AUTOMATIONDIRECT.com

DL205 PLC User Manual

Manual Number: D2-USER-M

WARNING

Thank you for purchasing automation equipment from **Automationdirect.com®**, doing business as, **AutomationDirect**. We want your new automation equipment to operate safely. Anyone who installs or uses this equipment should read this publication (and any other relevant publications) before installing or operating the equipment.

To minimize the risk of potential safety problems, you should follow all applicable local and national codes that regulate the installation and operation of your equipment. These codes vary from area to area and usually change with time. It is your responsibility to determine which codes should be followed, and to verify that the equipment, installation, and operation is in compliance with the latest revision of these codes.

At a minimum, you should follow all applicable sections of the National Fire Code, National Electrical Code, and the codes of the National Electrical Manufacturer's Association (NEMA). There may be local regulatory or government offices that can also help determine which codes and standards are necessary for safe installation and operation.

Equipment damage or serious injury to personnel can result from the failure to follow all applicable codes and standards. We do not guarantee the products described in this publication are suitable for your particular application, nor do we assume any responsibility for your product design, installation, or operation.

Our products are not fault-tolerant and are not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the product could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). **AutomationDirect** specifically disclaims any expressed or implied warranty of fitness for High Risk Activities.

For additional warranty and safety information, see the Terms and Conditions section of our catalog. If you have any questions concerning the installation or operation of this equipment, or if you need additional information, please call us at 1-770-844-4200.

This publication is based on information that was available at the time it was printed. At **AutomationDirect** we constantly strive to improve our products and services, so we reserve the right to make changes to the products and/or publications at any time without notice and without any obligation. This publication may also discuss features that may not be available in certain revisions of the product.

Trademarks

This publication may contain references to products produced and/or offered by other companies. The product and company names may be trademarked and are the sole property of their respective owners. **AutomationDirect** disclaims any proprietary interest in the marks and names of others.

ADVERTENCIA

Gracias por comprar equipo de automatización de **Automationdirect.com**®. Deseamos que su nuevo equipo de automatización opere de manera segura. Cualquier persona que instale o use este equipo debe leer esta publicación (y cualquier otra publicación pertinente) antes de instalar u operar el equipo.

Para reducir al mínimo el riesgo debido a problemas de seguridad, debe seguir todos los códigos de seguridad locales o nacionales aplicables que regulan la instalación y operación de su equipo. Estos códigos varian de área en área y usualmente cambian con el tiempo. Es su responsabilidad determinar cuales códigos deben ser seguidos y verificar que el equipo, instalación y operación estén en cumplimiento con la revisión mas reciente de estos códigos.

Como mínimo, debe seguir las secciones aplicables del Código Nacional de Incendio, Código Nacional Eléctrico, y los códigos de (NEMA) la Asociación Nacional de Fabricantes Eléctricos de USA. Puede haber oficinas de normas locales o del gobierno que pueden ayudar a determinar cuales códigos y normas son necesarios para una instalación y operación segura.

Si no se siguen todos los códigos y normas aplicables, puede resultar en daños al equipo o lesiones serias a personas. No garantizamos los productos descritos en esta publicación para ser adecuados para su aplicación en particular, ni asumimos ninguna responsabilidad por el diseño de su producto, la instalación u operación.

Nuestros productos no son tolerantes a fallas y no han sido diseñados, fabricados o intencionados para uso o reventa como equipo de control en línea en ambientes peligrosos que requieren una ejecución sin fallas, tales como operación en instalaciones nucleares, sistemas de navegación aérea, o de comunicación, control de tráfico aéreo, máquinas de soporte de vida o sistemas de armamentos en las cuales la falla del producto puede resultar directamente en muerte, heridas personales, o daños físicos o ambientales severos ("Actividades de Alto Riesgo"). **Automationdirect.com** específicamente rechaza cualquier garantía ya sea expresada o implicada para actividades de alto riesgo.

Para información adicional acerca de garantía e información de seguridad, vea la sección de Términos y Condiciones de nuestro catálogo. Si tiene alguna pregunta sobre instalación u operación de este equipo, o si necesita información adicional, por favor llámenos al número 1-770-844-4200 en Estados Unidos.

Esta publicación está basada en la información disponible al momento de impresión. En **Automationdirect.com** nos esforzamos constantemente para mejorar nuestros productos y servicios, así que nos reservamos el derecho de hacer cambios al producto y/o a las publicaciones en cualquier momento sin notificación y sin ninguna obligación. Esta publicación también puede discutir características que no estén disponibles en ciertas revisiones del producto.

Marcas Registradas

Esta publicación puede contener referencias a productos producidos y/u ofrecidos por otras compañías. Los nombres de las compañías y productos pueden tener marcas registradas y son propiedad única de sus respectivos dueños. Automationdirect.com, renuncia cualquier interés propietario en las marcas y nombres de otros.

AVERTISSEMENT

Nous vous remercions d'avoir acheté l'équipement d'automatisation de **Automationdirect.com®**, en faisant des affaires comme, **AutomationDirect**. Nous tenons à ce que votre nouvel équipement d'automatisation fonctionne en toute sécurité. Toute personne qui installe ou utilise cet équipement doit lire la présente publication (et toutes les autres publications pertinentes) avant de l'installer ou de l'utiliser.

Afin de réduire au minimum le risque d'éventuels problèmes de sécurité, vous devez respecter tous les codes locaux et nationaux applicables régissant l'installation et le fonctionnement de votre équipement. Ces codes diffèrent d'une région à l'autre et, habituellement, évoluent au fil du temps. Il vous incombe de déterminer les codes à respecter et de vous assurer que l'équipement, l'installation et le fonctionnement sont conformes aux exigences de la version la plus récente de ces codes.

Vous devez, à tout le moins, respecter toutes les sections applicables du Code national de prévention des incendies, du Code national de l'électricité et des codes de la National Electrical Manufacturer's Association (NEMA). Des organismes de réglementation ou des services gouvernementaux locaux peuvent également vous aider à déterminer les codes ainsi que les normes à respecter pour assurer une installation et un fonctionnement sûrs.

L'omission de respecter la totalité des codes et des normes applicables peut entraîner des dommages à l'équipement ou causer de graves blessures au personnel. Nous ne garantissons pas que les produits décrits dans cette publication conviennent à votre application particulière et nous n'assumons aucune responsabilité à l'égard de la conception, de l'installation ou du fonctionnement de votre produit.

Nos produits ne sont pas insensibles aux défaillances et ne sont ni conçus ni fabriqués pour l'utilisation ou la revente en tant qu'équipement de commande en ligne dans des environnements dangereux nécessitant une sécurité absolue, par exemple, l'exploitation d'installations nucléaires, les systèmes de navigation aérienne ou de communication, le contrôle de la circulation aérienne, les équipements de survie ou les systèmes d'armes, pour lesquels la défaillance du produit peut provoquer la mort, des blessures corporelles ou de graves dommages matériels ou environnementaux («activités à risque élevé»). La société **AutomationDirect** nie toute garantie expresse ou implicite d'aptitude à l'emploi en ce qui a trait aux activités à risque élevé.

Pour des renseignements additionnels touchant la garantie et la sécurité, veuillez consulter la section Modalités et conditions de notre documentation. Si vous avez des questions au sujet de l'installation ou du fonctionnement de cet équipement, ou encore si vous avez besoin de renseignements supplémentaires, n'hésitez pas à nous téléphoner au 1-770-844-4200.

Cette publication s'appuie sur l'information qui était disponible au moment de l'impression. À la société **AutomationDirect**, nous nous efforçons constamment d'améliorer nos produits et services. C'est pourquoi nous nous réservons le droit d'apporter des modifications aux produits ou aux publications en tout temps, sans préavis ni quelque obligation que ce soit. La présente publication peut aussi porter sur des caractéristiques susceptibles de ne pas être offertes dans certaines versions révisées du produit.

Marques de commerce

La présente publication peut contenir des références à des produits fabriqués ou offerts par d'autres entreprises. Les désignations des produits et des entreprises peuvent être des marques de commerce et appartiennent exclusivement à leurs propriétaires respectifs. **AutomationDirect** nie tout intérêt dans les autres marques et désignations.

DL205 PLC USER MANUAL

AUTOMATIONDIRECT.com

Please include the Manual Number and the Manual Issue, both shown below, when communicating with Technical Support regarding this publication.

Manual Number: D2-USER-M
Issue: 5th Edition, Rev. A
Issue Date: 11/20

Publication History		
Issue	**Date**	**Description of Changes**
1st Edition	1/94	Original edition
Rev. A	9/95	Minor corrections
2nd Edition	6/97	Added DL250, downsized manual
Rev. A	5/98	Minor corrections
Rev. B	7/99	Added torque specs for base and I/O
Rev. C	11/99	Minor corrections
Rev. D	3/00	Added new PID features, minor corrections
Rev. E	11/00	Added CE information, minor corrections
Rev. F	11/01	Added surge protection info, corrected RLL and DRUM instructions, minor corrections
3rd Edition	6/02	Added DL250–1 and DL260 CPUs, local expansion I/O, ASCII and MODBUS instructions, split manual into two volumes
Rev. A	8/03	Extensive corrections and additions
4th Edition	11/08	Changed publishing software resulting in change of appearance, addition of IBox instructions, changes to PID chapter, added info for ERM and EBC modules, other changes as necessary
Rev. A	4/10	Extensive corrections and additions
Rev. B	2/13	Corrected number of memory registers needed in the print message instruction. Added new transient suppression for inductive loads to Chapter 2. Added H2-CTRIO2 and H2-ERM100 references.
Rev. C	4/17	Minor corrections with general updates. ECEMAIL Decimal Status Codes added, Chapter 5
Rev. D	10/17	Minor corrections with general updates
5th Edition	12/18	Added DL262 CPU, extensive additions and corrections
Rev. A	11/20	Added lower baud rate values for comm port, pg 3-9

DL205 User Manual
Table of Contents

Chapter 1 - Getting Started

Chapter 2 - Installation , Wiring and Specifications

Chapter 3 - CPU Specifications and Operations

Chapter 4 - System Design and Configuration

Chapter 5 - RLL and Intelligent Box Instructions

Chapter 6 - Drum Instruction Programming (D2-250-1, D2-260 and D2-262 Only)

Chapter 8 - PID Loop Operation (D2-250-1/D2-260/D2-262)

Chapter 9 - Maintenance and Trouble Shooting

Appendix A - Auxiliary Functions

Appendix D - Special Relays

Appendix J - D2-262 CPU

Getting Started

In This Chapter...

Introduction

The Purpose of this Manual

Thank you for purchasing our DL205 family of products. This manual shows you how to install, program, and maintain the equipment. It also helps you understand how to interface them to other devices in a control system.

This manual contains important information for personnel who will install DL205 PLCs and components and for the PLC programmer. If you understand PLC systems, our manuals will provide all the information you need to start and keep your system up and running.

Where to Begin

If you already understand PLCs please read Chapter 2, "Installation, Wiring, and Specifications," and proceed on to other chapters as needed. Keep this manual handy for reference when you have questions. If you are a new DL205 customer, we suggest you read this manual completely to understand the wide variety of features in the DL205 family of products. We believe you will be pleasantly surprised with how much you can accomplish with our products.

Supplemental Manuals

If you have purchased operator interfaces or *Direct*SOFT, you will need to supplement this manual with the manuals that are written for those products.

Technical Support

We strive to make our manuals the best in the industry. We rely on your feedback to let us know if we are reaching our goal. If you cannot find the solution to your particular application, or, if for any reason you need technical assistance, please call us at:

770–844–4200

Our technical support group will work with you to answer your questions. They are available Monday through Friday from 9:00 A.M. to 6:00 P.M. Eastern Time. We also encourage you to visit our web site where you can find technical and non-technical information about our products and our company.

http://www.automationdirect.com

If you have a comment, question or suggestion about any of our products, services, or manuals, please fill out and return the 'Suggestions' card included with this manual.

Conventions Used

When you see the "notepad" icon in the left–hand margin, the paragraph to its immediate right will be a special note.

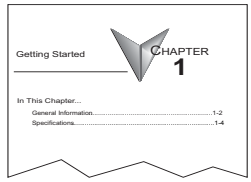
The word **NOTE** in boldface will mark the beginning of the text.

When you see the "exclamation mark" icon in the left–hand margin, the paragraph to its immediate right will be a warning. This information could prevent injury, loss of property, or even death (in extreme cases).

The word **WARNING** in boldface will mark the beginning of the text.

Key Topics for Each Chapter

The beginning of each chapter will list the key topics that can be found in that chapter.



DL205 System Components

The DL205 family is a versatile product line that provides a wide variety of features in an extremely compact package. The CPUs are small, but offer many instructions normally only found in larger, more expensive systems. The modular design also offers more flexibility in the fast moving industry of control systems. The following is a summary of the major DL205 system components.

CPUs

This product line includes five feature-enhanced CPUs: the D2-230, D2-240, D2-250–1, D2-260 and D2-262 (the D2-230, D2-240, D2-250 CPUs have been retired). All CPUs include built-in communication ports. Each CPU offers a large amount of program memory, a substantial instruction set and advanced diagnostics. The D2-250–1 features drum timers, floating–point math, 4 built-in PID loops with automatic tuning and 2 bases of local expansion capability.

In addition to the D2-250-1 features the D2-260 and D2-262 feature ASCII IN/OUT and extended MODBUS communications, table and trigonometric instructions, 16 PID loops with auto tuning and up to 4 bases of local expansion. Further details of these CPU features and more are covered in Chapter 3, CPU Specifications and Operation.

Bases

Four base sizes are available: 3, 4, 6 and 9 slot. The DL205 PLCs use bases that can be expanded. The part numbers for these bases end with –1. These bases have a connector for local expansion located on the right end of the base. They can serve as local bases, local expansion and remote I/O configurations. All bases include a built-in power supply. The bases with the –1 suffix can replace existing bases without a suffix if expansion is required.

I/O Configuration

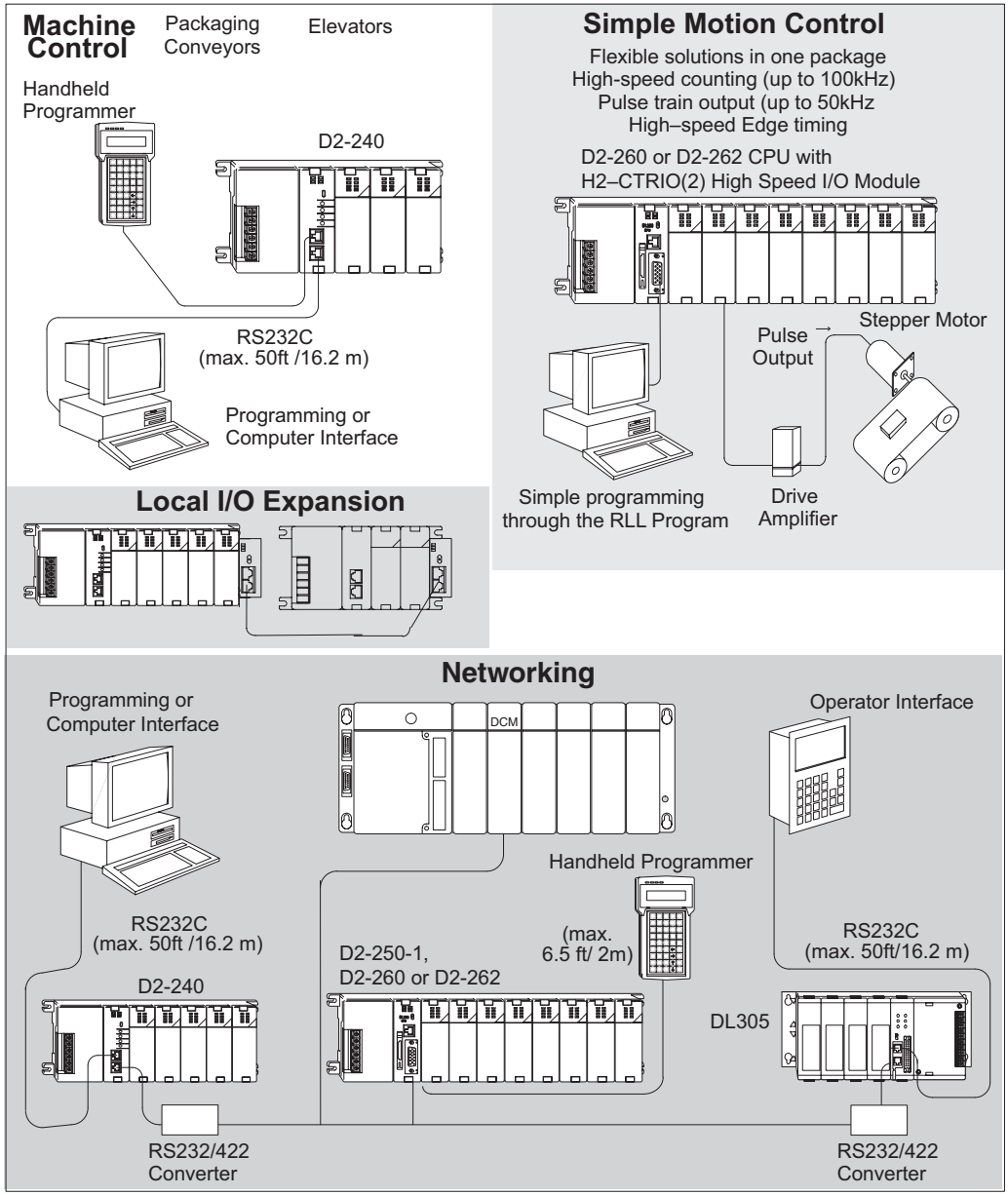
The D2-250–1 can support up to 768 local I/O points with up to two expansion bases. The D2-260 and D2-262 support up to 1280 local I/O points with up to four expansion bases. These points can be assigned as input or output points. The D2-250–1, D2-260 and D2-262 systems can also be expanded by adding remote I/O points. The D2-250–1, D2-260 and D2-262 provide a built–in master for remote I/O networks. The I/O module configuration are explained in Chapter 4, System Design and Configuration.

I/O Modules

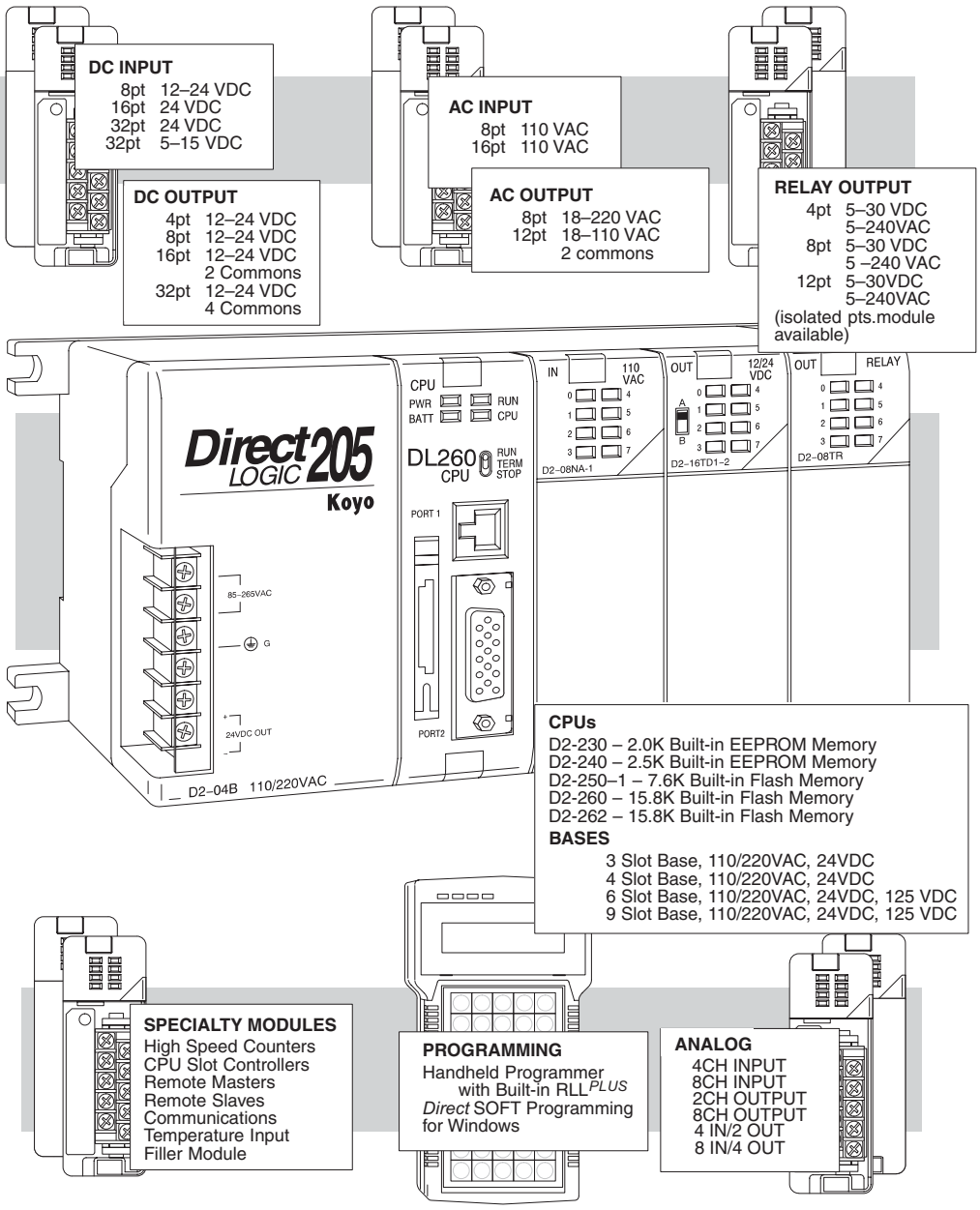
The DL205 series has a large range of I/O modules to accommodate your automation needs. A complete range of discrete modules which support 24VDC, 110/220 VAC and up to 10A relay outputs (subject to derating) are offered. The analog modules provide 12- and 16-bit resolution and several selections of input and output signal ranges (including bipolar). Several specialty and communications modules are also available.

DL205 System Diagrams

The diagram below shows the major components and configurations of the DL205 system. The next two pages show specific components for building your system.



*Direct*LOGIC DL205 Family



Programming Methods

Two programming methods are available for the DL205 CPUs: Relay Ladder Logic (RLL) and RLL^{PLUS} (Stage Programming). Both the *Direct*SOFT programming package and the handheld programmer support RLL and Stage programming.

DirectSOFT Programming for Windows

The DL205 can be programmed with one of the most advanced programming packages in the industry —*Direct*SOFT. *Direct*SOFT is a Windows-based software package that supports many Windows features you already know, such as cut and paste between applications, point and click editing, viewing and editing multiple application programs at the same time, etc. *Direct*SOFT universally supports the *Direct*LOGIC CPU families. This means you can use the same *Direct*SOFT programming software to program any CPU that is in the DL05, DL06, DL205, DL305, DL405 series. A separate manual discusses the *Direct*SOFT programming software which is included with your software package.

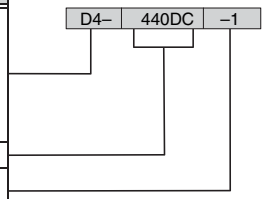
Handheld Programmer

All DL205 CPUs have a built-in programming port for use with the handheld programmer (D2–HPP). The handheld programmer can be used to create, modify and debug your application program. A separate manual that discusses the DL205 Handheld Programmer is available for download at **http://www.automationdirect.com**. Just search for "Handheld Programmer" and follow links.

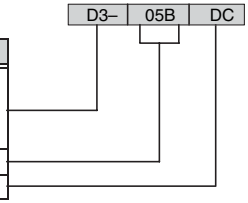
*Direct*LOGIC™ Part Numbering System

As you examine this manual, you will notice there are many different products available. Sometimes it is difficult to remember the specifications for any given product. However, if you take a few minutes to understand the numbering system, it may save you some time and confusion. The charts below show how the part numbering systems work for each product category. Part numbers for accessory items such as cables, batteries, memory cartridges, etc, are typically an abbreviation of the description for the item.

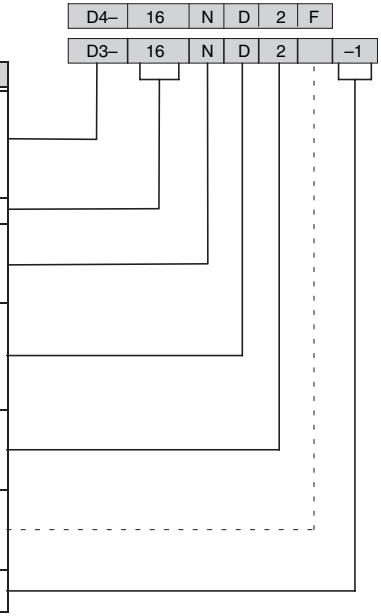
CPUs **Specialty CPUs**	
DL05/06 Product family DL105 Product family DL205 Product family DL305 Product family DL405 Product family	D0/F0 D1/F1 D2/F2 D3/F3 D4/F4
Class of CPU / Abbreviation	230...,330...,430...
Denotes a differentiation between similar modules	–1, –2, –3, –4



Bases	
DL205 Product family DL305 Product family DL405 Product family	D2/F2 D3/F3 D4/F4
Number of slots	##B
Type of Base	DC or empty



Discrete I/O	
DL05/06 Product family DL205 Product family DL305 Product family DL405 Product family	D0/F0 D2/F2 D3/F3 D4/F4
Number of points	04/08/12/16/32/64
Input Output Combination	N T C
AC DC Either Relay	A D E R
Current Sinking Current Sourcing Current Sinking/Sourcing	1 2 3
High Current Isolation Fast I/O	H S F
Denotes a differentiation between similar modules	–1, –2, –3, –4

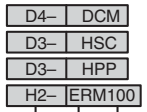


Analog I/O	
DL05/06 Product family	D0/F0
DL205 Product family	D2/F2
DL305 Product family	D3/F3
DL405 Product family	D4/F4
Number of channels	02/04/08/16
Input (Analog to Digital)	AD
Output (Digital to Analog)	DA
Combination	AND
Isolated	S
Denotes a differentiation between Similar modules	–1, –2, –3, –4

Alternate example of Analog I/O using abbreviations

note: –n indicates thermocouple type such as: J, K, T, R, S or E

Communication and Networking Special I/O and Devices Programming	
DL205 Product family	D2/F2/H2
DL305 Product family	D3/F3
DL405 Product family	D4/F4/H4
Name Abbreviation	see example



(Data Communication Module)
(High Speed Counter)
(RLL PLUS Handheld Programmer)
(Ethernet Remote I/O Master)

CoProcessors and ASCII BASIC Modules	
DL205 Product family	D2/F2
DL305 Product family	D3/F3
DL405 Product family	D4/F4
CoProcessor	CP
ASCII BASIC	AB
64K memory	64
128K memory	128
512K memory	512
Radio modem	R
Telephone modem	T

Quick Start for PLC Validation and Programming

If you have experience using PLCs, or want to set up a quick example, this section is what you want to use. This example is not intended to explain everything needed to start up your system. It is only intended to provide a general picture of what is needed to get your system powered up.

Step 1: Unpack the DL205 Equipment

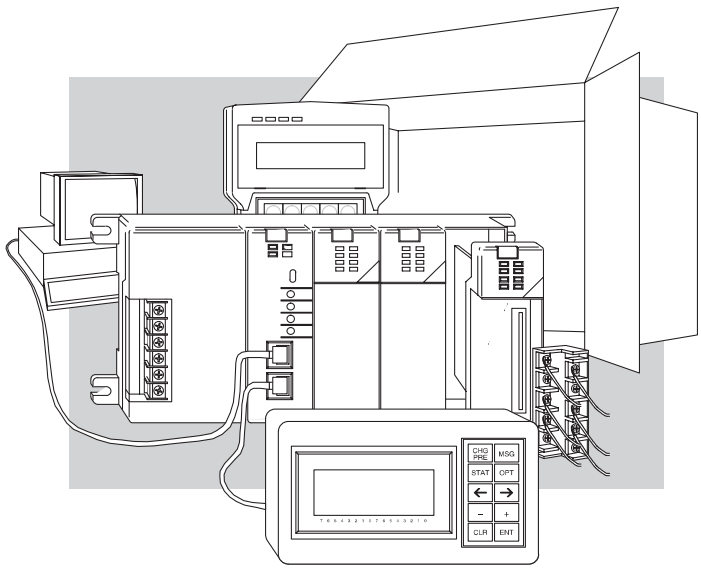
Unpack the DL205 equipment and verify you have the parts necessary to build this demonstration system. The minimum parts needed are as follows:

- Base
- CPU
- A discrete input module such as a D2–16ND3–2 or a F2–08SIM input simulator module
- A discrete output module such as a D2–16TD1–2
- *Power cord
- *Hookup wire
- *One or more toggle switches (if not using the input simulator module)
- *A screwdriver, blade or Phillips type

*These items are not supplied with your PLC.

You will need at least one of the following programming options:

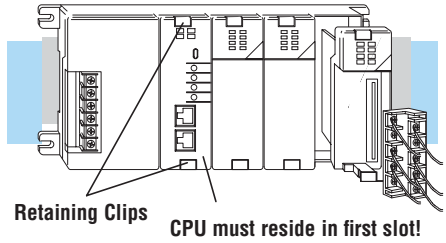
- ***Direct***SOFT Programming Software, ***Direct***SOFT Manual, and a programming cable (connects the CPU to a personal computer), or
- D2–HPP Handheld Programmer and the Handheld Programmer Manual.



Step 2: Install the CPU and I/O Modules

Insert the CPU and I/O into the base. The CPU must be inserted into the first slot of the base (next to the power supply).

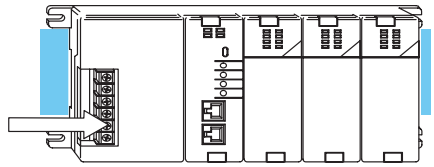
- Each unit has a plastic retaining clip at the top and bottom. Slide the retainer clips to the out position before installing the module.
- With the unit square to the base, slide it in using the upper and lower guides.
- Gently push the unit back until it is firmly seated in the backplane.
- Secure the unit to the base by pushing in the retainer clips.



Placement of discrete, analog and relay modules is not critical and may go in any slot in any base; however, for this example, install the output module in the slot next to the CPU and the input module in the next slot. Limiting factors for other types of modules are discussed in Chapter 4, System Design and Configuration. You must also make sure you do not exceed the power budget for each base in your system configuration. Power budgeting is also discussed in Chapter 4.

Step 3: Remove Terminal Strip Access Cover

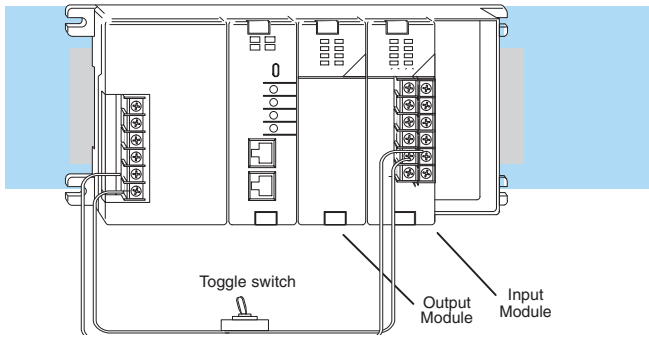
Remove the terminal strip cover. It is a small strip of clear plastic that is located on the base power supply.



Step 4: Add I/O Simulation

To finish this quick start exercise or study other examples in this manual, you will need to install an input simulator module (or wire an input switch as shown below), and add an output module. Using an input simulator is the quickest way to get physical inputs for checking out the system or a new program. To monitor output status, any discrete output module will work.

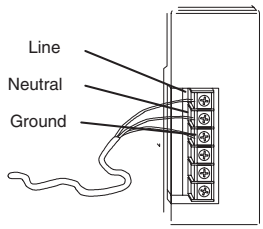
Wire the switches or other field devices prior to applying power to the system to ensure a point is not accidentally turned on during the wiring operation. This example uses DC input



and output modules. Wire the input module, X0, to the toggle switch and 24VDC auxiliary power supply on the CPU terminal strip as shown. Chapter 2, Installation, Wiring, and Specifications, provides a list of I/O wiring guidelines.

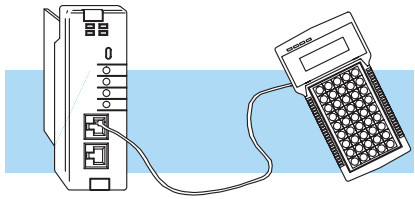
Step 5: Connect the Power Wiring

Connect the wires as shown. Observe all precautions stated earlier in this manual. For details on wiring see Chapter 2 Installation, Wiring, and Specifications. When the wiring is complete, replace the CPU and module covers. Do not apply power at this time.



Step 6: Connect the Programmer

Either connect the programming cable connected to a computer loaded with *Direct*SOFT Programming Software or a D2-HPP Handhelд Programmer (comes with programming cable) to the top port of the CPU.



Step 7: Switch On the System Power

Apply power to the system and ensure the PWR indicator on the CPU is on. If not, remove power from the system, check all wiring and refer to the troubleshooting section in Chapter 9 for assistance.

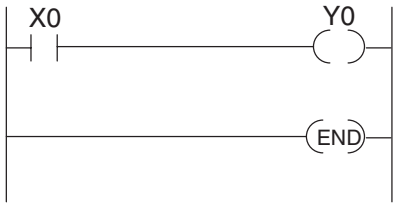
Step 8: Enter the Program

Slide the switch on the CPU to the STOP position (D2-250–1, D2-260 and D2-262 CPUs) and then back to the TERM position. This puts the CPU in the program mode and allows access to the CPU program. Edit a *Direct*SOFT program using the relay ladder diagram below and load it into the PLC. If using an HPP, the PGM indicator should be illuminated on the HPP. Enter the following keystrokes on the HPP:

NOTE: *It is not necessary for you to configure the I/O for this system since the DL205 CPUs automatically examine any installed modules and establish the correct configuration.*

Handheld Program Keystrokes

$ STR	→	B 1	ENT
GX OUT	→	C 2	ENT

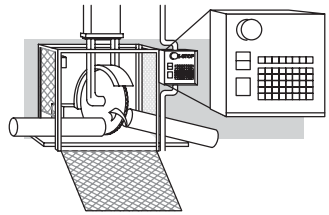


After entering the example program put the CPU in the RUN mode with *Direct*SOFT or after entering the program using the HPP, slide the switch from the TERM position to the RUN position and back to TERM. The RUN indicator on the CPU will come on indicating the CPU has entered the run mode. If not, repeat Step 8 ensuring the program is entered properly or refer to the troubleshooting guide in Chapter 9.

During Run mode operation, the output status indicator "0" on the output module should reflect the switch status. When the switch is on, the output should be on.

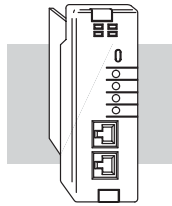
Steps to Designing a Successful System

Step 1: Review the Installation Guidelines



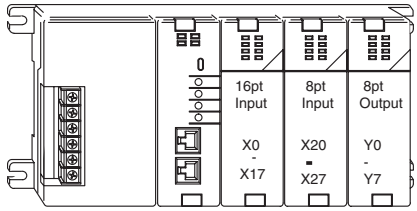
Always make safety your first priority in any system application. Chapter 2 provides several guidelines that will help provide a safer, more reliable system. This chapter also includes wiring guidelines for the various system components.

Step 2: Understand the CPU Set-up Procedures



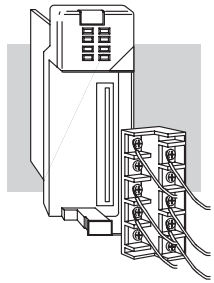
The CPU is the heart of your automation system and is explained in Chapter 3. Make sure you take time to understand the various features and set-up requirements.

Step 3: Understand the I/O System Configurations



It is important to understand how your local I/O system can be configured. It is also important to understand how the system Power Budget is calculated. This can affect your I/O placement and/or configuration options. See Chapter 4 for more information.

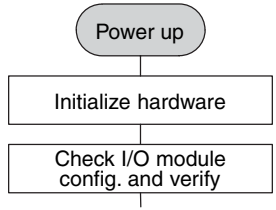
Step 4: Determine the I/O Module Specifications and Wiring Characteristics



Many different I/O modules are available with the DL205 system. Chapter 2 provides the specifications and wiring diagrams for the discrete I/O modules.

***NOTE**: Analog and specialty modules have their own manuals and are not included in this manual.*

Step 5: Understand the System Operation



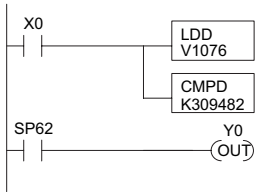
Before you begin to enter a program, it is very helpful to understand how the DL205 system processes information. This involves not only program execution steps, but also involves the various modes of operation and memory layout characteristics. See Chapter 3 for more information.

Step 6: Review the Programming Concepts

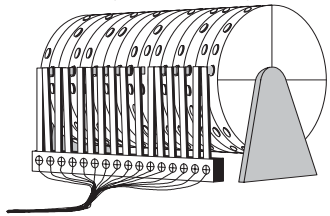
The DL205 provides four main approaches to solving the application program, including the PID loop task depicted in the next figure.

- RLL diagram style programming is the best tool for solving boolean logic and general CPU register/ accumulator manipulation. It includes dozens of instructions, which will augment drums, stages and loops.
- The D2-250-1, D2-260 and D2-262 have four timer/event drum types, each with up to 16 steps. They offer both time and/or event-based step transitions. Drums are best for a repetitive process based on a single series of steps.
- Stage programming, called RLL*PLUS*, is based on state-transition diagrams. Stages divide the ladder program into sections which correspond to the states in a flow chart of your process.
- The D2-260 and D2-262 CPU PID loop operation uses setup tables to configure 16 loops. The D2-250-1 PID loop operation uses setup to configure 4 loops. Features include: auto tuning, alarms, SP ramp/soak generation and more.

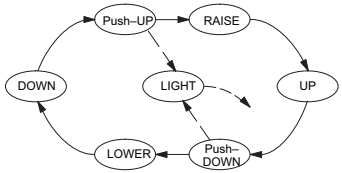
Standard RLL Programming
(see Chapter 5)



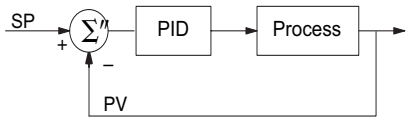
Timer/Event Drum Sequencer
(see Chapter 6)



Stage Programming
(see Chapter 7)

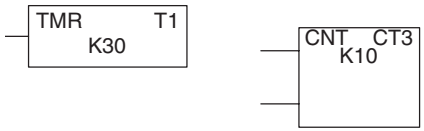


PID Loop Operation
(see Chapter 8)



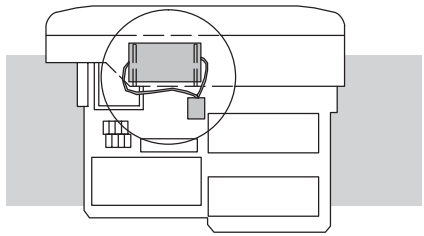
Step 7: Choose the Instructions

Once you have installed the system and understand the theory of operation, you can choose from one of the most powerful instruction sets available.



Step 8: Understand the Maintenance and Troubleshooting Procedures

Equipment failures can occur at any time. Switches fail, batteries need to be replaced, etc. In most cases, the majority of the troubleshooting and maintenance time is spent trying to locate the problem. The DL205 system has many built-in features that help you quickly identify problems. Refer to Chapter 9 for diagnostics.



Installation, Wiring and Specifications

In This Chapter...

Safety Guidelines

NOTE: Products with CE marks *perform their required functions safely and adhere to relevant standards as specified by CE directives, provided they are used according to their intended purpose and that the instructions in this manual are adhered to. The protection provided by the equipment may be impaired if this equipment is used in a manner not specified in this manual. A listing of our international affiliates is available on our Web site: http://www.automationdirect.com*

WARNING: Providing a safe operating environment for personnel and equipment is your responsibility and should be your primary goal during system planning and installation. Automation systems can fail and may result in situations that can cause serious injury to personnel and/or damage equipment. Do not rely on the automation system alone to provide a safe operating environment. Sufficient emergency circuits should be provided to stop either partially or totally the operation of the PLC or the controlled machine or process. These circuits should be routed outside the PLC in the event of controller failure, so that independent and rapid shutdown is available. Devices, such as "mushroom" switches or end of travel limit switches, should operate motor starter, solenoids, or other devices without being processed by the PLC. These emergency circuits should be designed using simple logic with a minimum number of highly reliable electromechanical components. Every automation application is different, so there may be special requirements for your particular application. Make sure to follow all national, state, and local government requirements for the proper installation and use of your equipment.

Plan for Safety

The best way to provide a safe operating environment is to make personnel and equipment safety part of the planning process. You should examine every aspect of the system to determine which areas are critical to operator or machine safety.

If you are not familiar with PLC system installation practices, or your company does not have established installation guidelines, you should obtain additional information from the following sources.

- NEMA — The National Electrical Manufacturers Association, located in Washington, D.C., publishes many different documents that discuss standards for industrial control systems. You can order these publications directly from NEMA. Some of these include:

  ICS 1, General Standards for Industrial Control and Systems

  ICS 3, Industrial Systems

  ICS 6, Enclosures for Industrial Control Systems

- NEC — The National Electrical Code provides regulations concerning the installation and use of various types of electrical equipment. Copies of the NEC Handbook can often be obtained from your local electrical equipment distributor or your local library.
- Local and State Agencies — many local and state governments have additional requirements above and beyond those described in the NEC Handbook. Check with your local Electrical Inspector or Fire Marshall office for information.

Three Levels of Protection

The publications mentioned provide many ideas and requirements for system safety. At a minimum, you should follow these regulations. Also, you should use the following techniques, which provide three levels of system control.

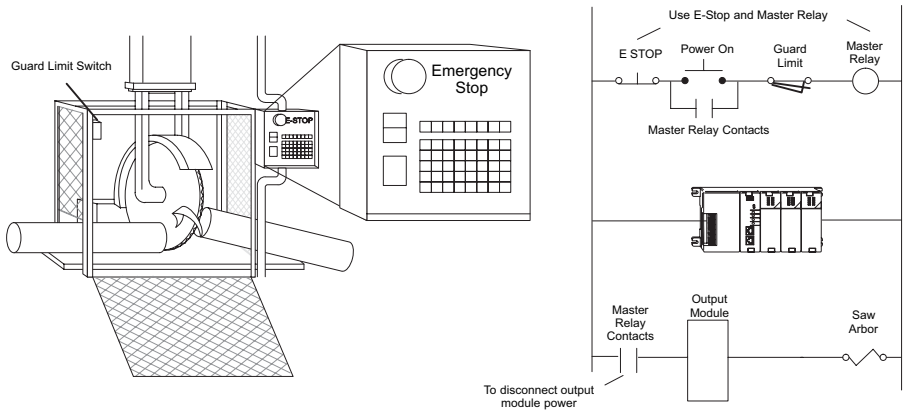
- Emergency stop switch for disconnecting system power
- Mechanical disconnect for output module power
- Orderly system shutdown sequence in the PLC control program

Emergency Stops

It is recommended that emergency stop circuits be incorporated into the system for every machine controlled by a PLC. For maximum safety in a PLC system, these circuits must not be wired into the controller, but should be hardwired external to the PLC. The emergency stop switches should be easily accessed by the operator and are generally wired into a master control relay (MCR) or a safety control relay (SCR) that will remove power from the PLC I/O system in an emergency.

MCRs and SCRs provide a convenient means for removing power from the I/O system during an emergency situation. By de-energizing an MCR (or SCR) coil, power to the input (optional) and output devices is removed. This event occurs when any emergency stop switch opens. However, the PLC continues to receive power and operate even though all its inputs and outputs are disabled.

The MCR circuit could be extended by placing a PLC fault relay (closed during normal PLC operation) in series with any other emergency stop conditions. This would cause the MCR circuit to drop the PLC I/O power in case of a PLC failure (memory error, I/O communications error, etc).



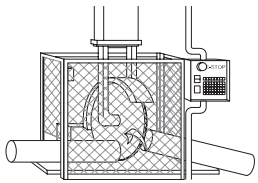
Emergency Power Disconnect

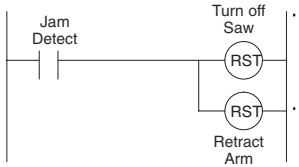
A properly rated emergency power disconnect should be used to power the PLC-controlled system as a means of removing the power from the entire control system. It may be necessary to install a capacitor across the disconnect to protect against a condition known as "outrush." This condition occurs when the output Triacs are turned off by powering off the disconnect, thus causing the energy stored in the inductive loads to seek the shortest distance to ground, which is often through the Triacs.

After an emergency shutdown or any other type of power interruption, there may be requirements that must be met before the PLC control program can be restarted. For example, there may be specific register values that must be established (or maintained from the state prior to the shutdown) before operations can resume. In this case, you may want to use retentive memory locations, or include constants in the control program to ensure a known starting point.

Orderly System Shutdown

Ideally, the first level of fault detection is the PLC control pr
which can identify machine problems. Certain shu
sequences should be performed. The types of proble
usually things such as jammed parts, etc., that do not pose a
personal injury or equipment damage.





WARNING: The control program *must not* be the only form of prot for any problems that may result in a risk of personal injury or equipment damage.

Class 1, Division 2, Approval

This equipment is suitable for use in Class 1, Zone 2, Division 2, groups A, B, C and D or non-hazardous locations only.

WARNING: Explosion Hazard! Substitution of components may impair suitability for Class 1, Division 2. Do not disconnect equipment unless power has been switched off or area is known to be non-hazardous.

WARNING: Explosion Hazard! Do not disconnect equipment unless power has been switched off or the area is known to be non-hazardous.

WARNING: All models used with connector accessories must use R/C (ECBT2) mating plug for all applicable models. All mating plugs shall have suitable ratings for device.

WARNING: This equipment is designed for use in Pollution Degree 2 environments (installed within an enclosure rated at least IP54).

WARNING: Transient suppression must be provided to prevent the rated voltage from being exceeded by 140%.

Mounting Guidelines

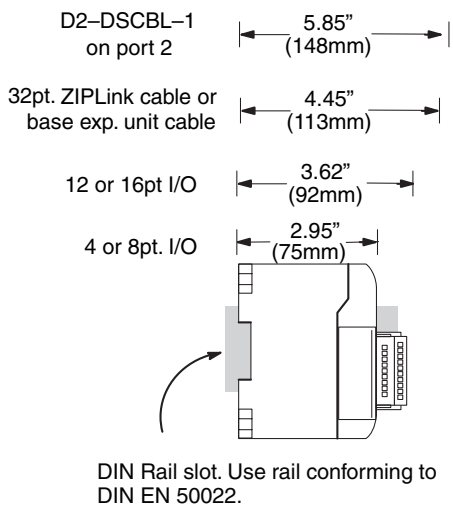
Before installing the PLC system, you will need to know the dimensions of the components considered. The diagrams on the following pages provide the component dimensions to use in defining your enclosure specifications. Remember to leave room for potential expansion.

NOTE: *If you are using other components in your system, refer to the appropriate manual to determine how those units can affect mounting dimensions.*

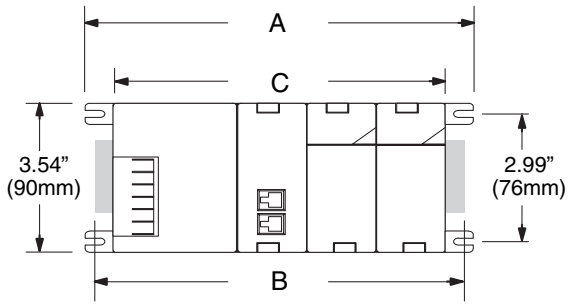
Base Dimensions

The following information shows the proper mounting dimensions. The height dimension is the same for all bases. The depth varies depending on your choice of I/O module. The length varies as the number of slots increase. Make sure you have followed the installation guidelines for proper spacing.

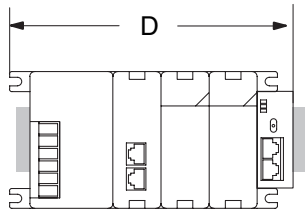
Mounting depths with:



DIN Rail slot. Use rail conforming to DIN EN 50022.



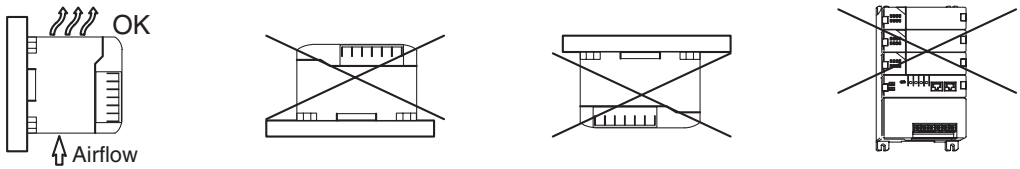
with D2–EM Expansion Unit



Base	A (Base Total Width)		B (Mounting Hole)		C (Component Width)		D (Width w/ Exp. Unit)	
	Inches	Millimeters	Inches	Millimeters	Inches	Millimeters	Inches	Millimeters
3-slot	6.77	172	6.41	163	5.8	148	7.24	184
4-slot	7.99	203	7.63	194	7.04	179	8.46	215
6-slot	10.43	265	10.07	256	9.48	241	10.90	277
9-slot	14.09	358	13.74	349	13.14	334	14.56	370

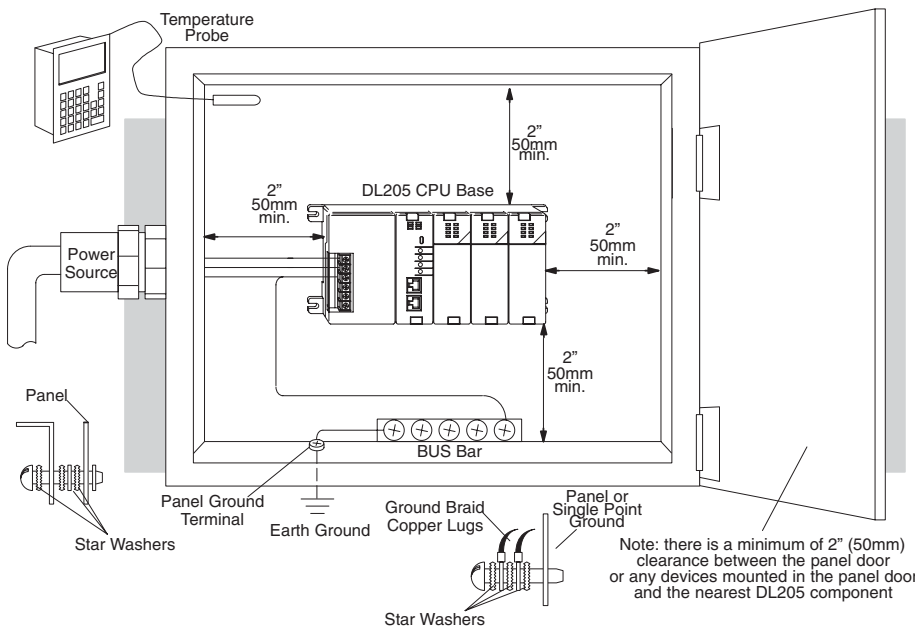
Panel Mounting and Layout

It is important to design your panel properly to help ensure the DL205 products operate within their environmental and electrical limits. The system installation should comply with all appropriate electrical codes and standards. It is important the system also conforms to the operating standards for the application to ensure proper performance. The diagrams below reference the items in the following list.



1. Mount the bases horizontally to provide proper ventilation.
2. If you place more than one base in a cabinet, there should be a minimum of 7.2" (183mm) between bases.
3. Provide a minimum clearance of 2" (50mm) between the base and all sides of the cabinet. There should also be at least 1.2" (30mm) of clearance between the base and any wiring ducts.
4. There must be a minimum of 2" (50mm) clearance between the panel door and the nearest DL205 component.

NOTE: *The cabinet configuration below is not suitable for EU installations. Refer to Appendix I, European Union Directives.*



5. The ground terminal on the DL205 base must be connected to a single point ground. Use copper stranded wire to achieve a low impedance. Copper eye lugs should be crimped and soldered to the ends of the stranded wire to ensure good surface contact. Remove anodized finishes and use copper lugs and star washers at termination points. A general rule is to achieve a 0.1 ohm of DC resistance between the DL205 base and the single point ground.
6. There must be a single point ground (i.e., copper bus bar) for all devices in the panel requiring an earth ground return. The single point of ground must be connected to the panel ground termination. The panel ground termination must be connected to earth ground. For this connection you should use #12 AWG stranded copper wire at a minimum. Minimum wire sizes, color coding, and general safety practices should comply with appropriate electrical codes and standards for your region. A good common ground reference (Earth ground) is essential for proper operation of the DL205. Methods of providing an adequate common ground reference include:
   - Installing a ground rod as close to the panel as possible.
   - Connection to incoming power system ground.
7. Properly evaluate any installations where the ambient temperature may approach the lower or upper limits of the specifications. Place a temperature probe in the panel, close the door and operate the system until the ambient temperature has stabilized. If the ambient temperature is not within the operating specification for the DL205 system, measures such as installing a cooling/heating source must be taken to get the ambient temperature within the DL205 operating specifications.
8. Device mounting bolts and ground braid termination bolts should be #10 copper bolts or equivalent. Tapped holes instead of nut–bolt arrangements should be used whenever possible. To ensure good contact on termination areas, impediments such as paint, coating or corrosion should be removed in the area of contact.
9. The DL205 system is designed to be powered by 110/220 VAC, 24VDC, or 125VDC normally available throughout an industrial environment. Electrical power in some areas where the PLCs are installed is not always stable and storms can cause power surges. Due to this, powerline filters are recommended for protecting the DL205 PLCs from power surges and EMI/RFI noise. The Automation Powerline Filter, for use with 120VAC and 240VAC, 1–5 Amps, is an excellent choice (can be located at www.automationdirect.com); however, you can use a filter of your choice. These units install easily between the power source and the PLC.

Enclosures

Your selection of a proper enclosure is important to ensure safe and proper operation of your DL205 system. Applications of DL205 systems vary and may require additional features. The minimum considerations for enclosures include:

- Conformance to electrical standards
- Protection from the elements in an industrial environment
- Common ground reference
- Maintenance of specified ambient temperature
- Access to equipment
- Security or restricted access
- Sufficient space for proper installation and maintenance of equipment

Environmental Specifications

The following table lists the environmental specifications that generally apply to the DL205 system (CPU, Bases, I/O Modules). The ranges that vary for the Handheld Programmer are noted at the bottom of this chart. I/O module operation may fluctuate depending on the ambient temperature and your application. Refer to the appropriate I/O module specifications for the temperature derating curves applying to specific modules.

Specification	Rating
Storage Temperature	–4°F to 158°F (–20°C to 70°C)
Ambient Operating Temperature*	32°F to 131°F (0°C to 55°C)
Ambient Humidity**	30% – 95% relative humidity (non–condensing)
Vibration Resistance	MIL STD 810C, Method 514.2
Shock Resistance	MIL STD 810C, Method 516.2
Noise Immunity	NEMA (ICS3–304)
Atmosphere	No corrosive gases

*** Operating temperature for the Handheld Programmer and the DV-1000 is 32° to 122°F (0° to 50°C) Storage temperature for the Handheld Programmer and the DV-1000 is - 4° to 158° F (- 20° to 70°C).**

**** Equipment will operate below 30% humidity. However, static electricity problems occur much more frequently at lower humidity levels. Make sure you take adequate precautions when you touch the equipment. Consider using ground straps, anti-static floor coverings, etc., if you use the equipment in low humidity environments.**

Power

The power source must be capable of supplying voltage and current complying with the base power supply specifications.

Specification	AC Powered Bases	24VDC Powered Bases	125VDC Powered Bases
Part Numbers	D2–03B–1, D2–04B–1, D2–06B–1 D2–09B–1	D2–03BDC1–1, D2–04BDC1–1, D2–06BDC1–1, D2–09BDC1–1	D2–06BDC2–1, D2–09BDC2–1
Input Voltage Range	100–240 VAC (+10%/ –15%) 50/60Hz	10.2 – 28.8 VDC (24VDC) with less than 10% ripple	104–240 VDC +10% –15%
Maximum Inrush Current	30A	10A	20A
Maximum Power	80VA	25W	30W
Voltage Withstand (dielectric)	1 minute @ 1500VAC between primary, secondary, and field ground		
Insulation Resistance	> 10 MΩ at 500VDC		
Auxiliary 24 VDC Output	20–28 VDC, less than 1V p-p 300mA max.	None	20–28 VDC, less than 1V p-p 300mA max.
Fusing (internal to base power supply)	Non–replaceable 2A @ 250V slow blow fuse	Non–replaceable 3.15 A @ 250V slow blow fuse	Non–replaceable 2A @ 250V slow blow fuse

Agency Approvals

Some applications require agency approvals. The DL205 are submitted to the following agency approvals:

- UL (Underwriters Laboratories, LLC)
- CE EMC (Electromagnetic Compatibility)
- cUL (Canadian Underwriters' Laboratories)

24VDC Power Bases

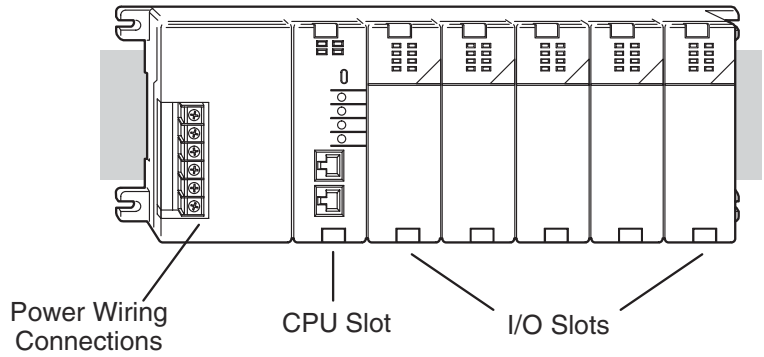
Follow these additional installation guidelines when installing D2-03BDC1-1, D2-04BDC1-1, D2-06BDC1-1 and D2-09BDC1-1 bases:

- Install these bases in compliance with the enclosure, mounting, spacing, and segregation requirements of the ultimate application.
- These bases must be used within their marked ratings.
- These bases are intended to be installed within an enclosure rated at least IP54.
- Provisions should be made to prevent the rated voltage being exceeded by transient disturbances of more than 40%.

Installing DL205 Bases

Choosing the Base Type

The DL205 system offers four different sizes of bases and three different power supply options. The following diagram shows an example of a 6-slot base.

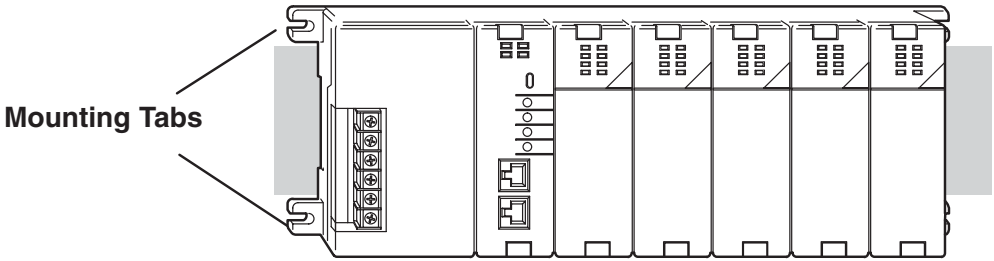


Your choice of base depends on three things:

- Number of I/O modules required
- Input power requirement (AC or DC power)
- Available power budget

Mounting the Base

All I/O configurations of the DL205 may use any of the base configurations. The bases are secured to the equipment panel or mounting location using four M4 screws in the corner tabs of the base. The full mounting dimensions are given in the previous section on Mounting Guidelines.



WARNING: To minimize the risk of electrical shock, personal injury, or equipment damage, always disconnect the system power before installing or removing any system component.

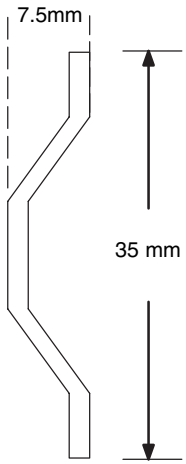
Using Mounting Rails

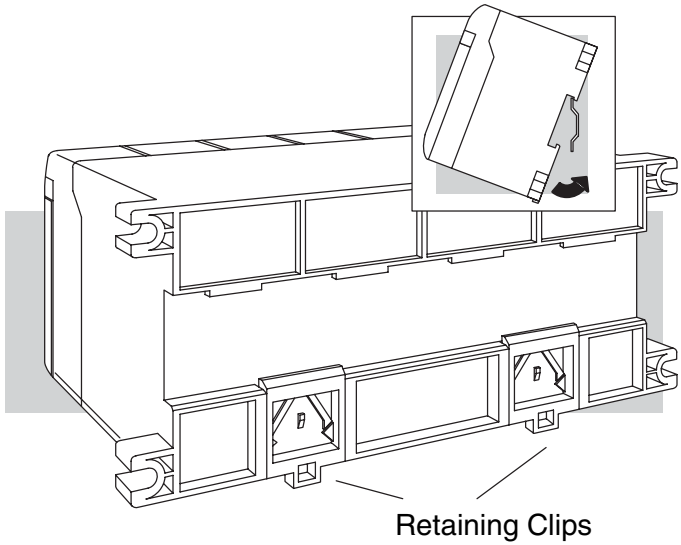
The DL205 bases can also be secured to the cabinet using mounting rails. You should use rails that conform to DIN EN standard 50 022. Refer to our catalog for a complete line of DIN-rail, DINnectors and DIN-rail mounted apparatus. These rails are approximately 35mm high, with a depth of 7.5 mm. If you mount the base on a rail, you should also consider using end brackets on each end of the rail. The end brackets help keep the base from sliding horizontally along the rail. This helps minimize the possibility of accidentally pulling the wiring loose.

If you examine the bottom of the base, you'll notice small retaining clips. To secure the base to a DIN-rail, place the base onto the rail and gently push up on the retaining clips. The clips lock the base onto the rail.

To remove the base, pull down on the retaining clips, lift up on the base slightly, and pull it away from the rail.

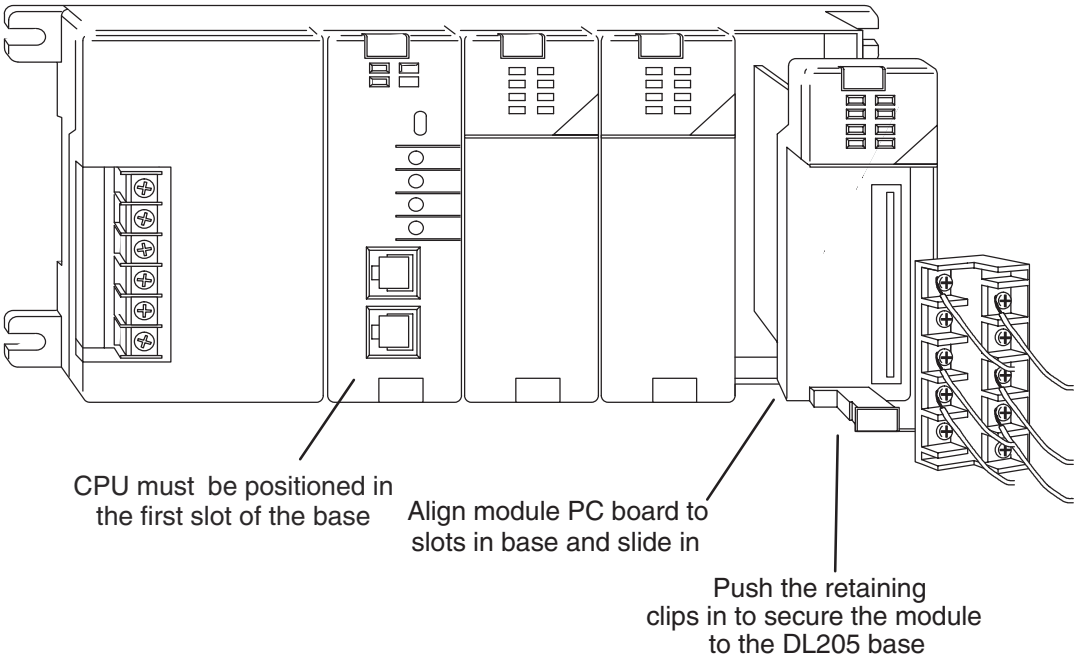
DIN Rail Dimensions





Installing Components in the Base

To insert components into the base: first slide the module retaining clips to the out position and align the PC board(s) of the module with the grooves on the top and bottom of the base. Push the module straight into the base until it is firmly seated in the backplane connector. Once the module is inserted into the base, push in the retaining clips to firmly secure the module to the base.



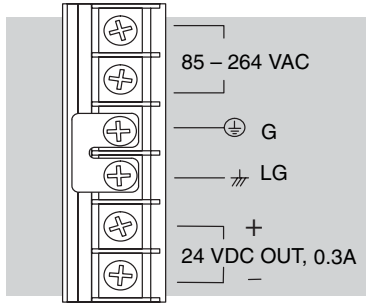
WARNING: Minimize the risk of electrical shock, personal injury, or equipment damage. Always disconnect the system power before installing or removing any system component.

Base Wiring Guidelines

Base Wiring

The diagrams show the terminal connections located on the power supply of the DL205 bases. The base terminals can accept up to 16 AWG. You may be able to use larger wiring depending on the type of wire used, but 16 AWG is the recommended size. Do not overtighten the connector screws; the recommended torque value is 7.81 lb·in (0.882 N·m).

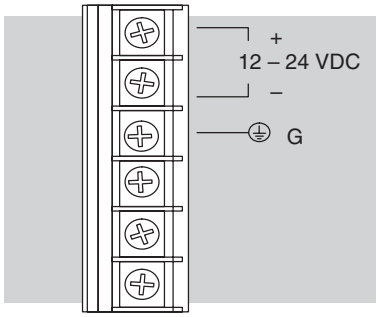
110/220 VAC Base Terminal Strip



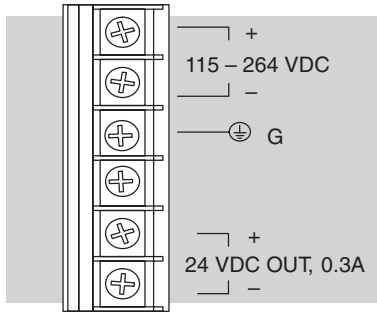
NOTE: *You can connect either a 115VAC or 220VAC supply to the AC terminals. Special wiring or jumpers are not required as with some of the other* ***Direct****LOGIC products.*

WARNING: Once the power wiring is connected, install the plastic protective cover. When the cover is removed there is a risk of electrical shock if you accidentally touch the wiring or wiring terminals.

12/24 VDC Base Terminal Strip



125 VDC Base Terminal Strip

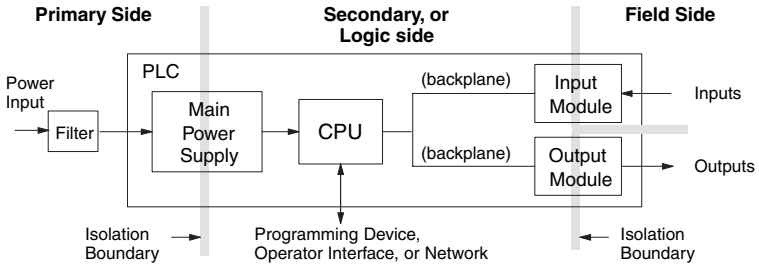


I/O Wiring Strategies

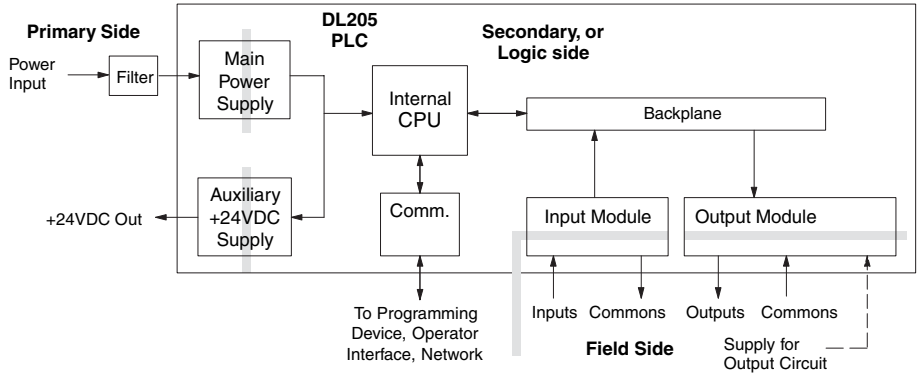
The DL205 PLC system is very flexible and will work in many different wiring configurations. By studying this section before actual installation, you can probably find the best wiring strategy for your application. This will help to lower system cost, wiring errors, and avoid safety problems.

PLC Isolation Boundaries

PLC circuitry is divided into three main regions separated by isolation boundaries, shown in the drawing below. Electrical isolation provides safety, so that a fault in one area does not damage another. A powerline filter will provide isolation between the power source and the power supply. A transformer in the power supply provides magnetic isolation between the primary and secondary sides. Opto-couplers provide optical isolation in Input and Output circuits. This isolates logic circuitry from the field side, where factory machinery connects. Note the discrete inputs are isolated from the discrete outputs, because each is isolated from the logic side. Isolation boundaries protect the operator interface (and the operator) from power input faults or field wiring faults. When wiring a PLC, it is extremely important to avoid making external connections that connect logic side circuits to any other.



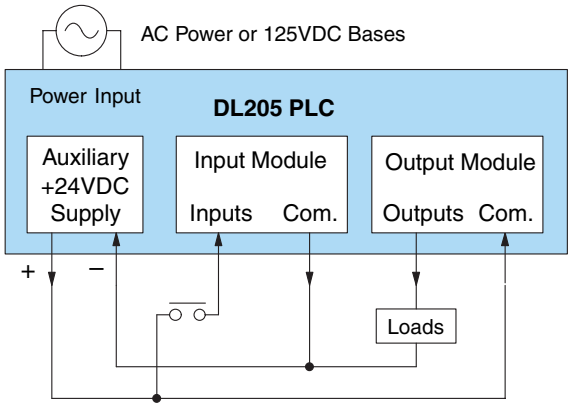
In addition to the basic circuits covered above, AC-powered and 125VDC bases include an auxiliary +24VDC power supply with its own isolation boundary. Since the supply output is isolated from the other three circuits, it can power input and/or output circuits!



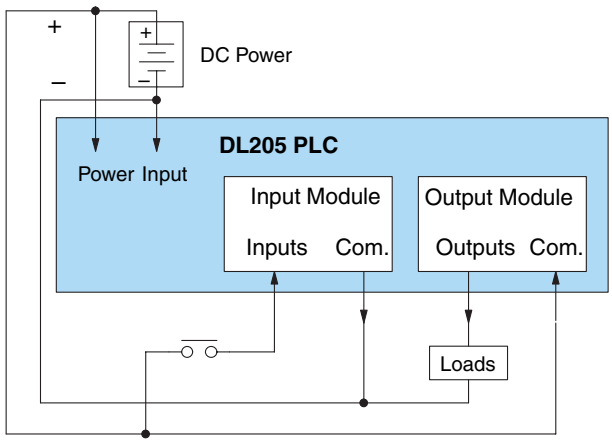
Powering I/O Circuits with the Auxiliary Supply

In some cases, using the built-in auxiliary +24VDC supply can result in a cost savings for your control system. It can power combined loads up to 300mA. Be careful not to exceed the current rating of the supply. If you are the system designer for your application, you may be able to select and design in field devices which can use the +24VDC auxiliary supply.

All AC powered and 125VDC DL205 bases feature the internal auxiliary supply. If input devices AND output loads need +24VDC power, the auxiliary supply may be able to power both circuits as shown in the following diagram.



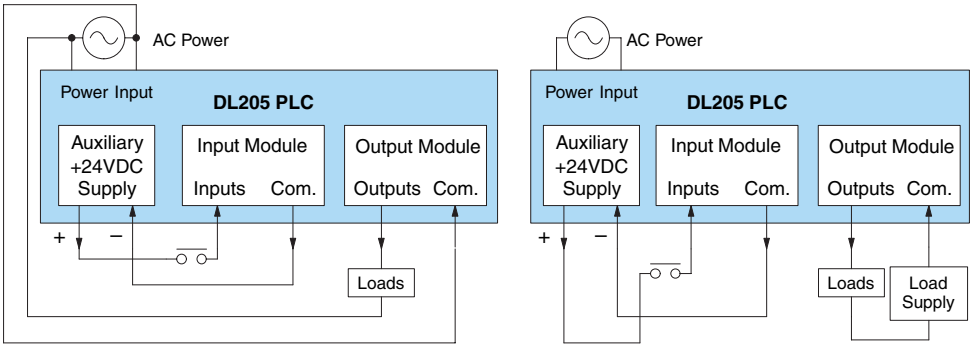
The 12/24 VDC-powered DL205 bases are designed for application environments in which low-voltage DC power is more readily available than AC. These include a wide range of battery–powered applications, such as remotely-located control, in vehicles, portable machines, etc. For this application type, all input devices and output loads typically use the same DC power source. Typical wiring for DC-powered applications is shown in the following diagram.



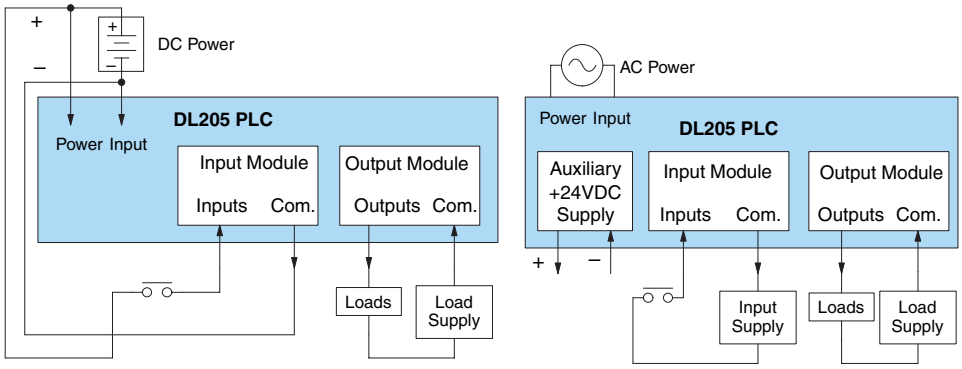
Powering I/O Circuits Using Separate Supplies

In most applications it will be necessary to power the input devices from one power source, and to power output loads from another source. Loads often require high-energy AC power, while input sensors use low-energy DC. If a machine operator is likely to come in close contact with input wiring, then safety reasons also require isolation from high-energy output circuits. It is most convenient if the loads can use the same power source as the PLC, and the input sensors can use the auxiliary supply, as shown to the left in the figure below.

If the loads cannot be powered from the PLC supply, then a separate supply must be used as shown to the right in the figure below.



Some applications will use the PLC external power source to also power the input circuit. This typically occurs on DC-powered PLCs, as shown in the drawing below to the left. The inputs share the PLC power source supply, while the outputs have their own separate supply. A worst-case scenario, from a cost and complexity viewpoint, is an application which requires separate power sources for the PLC, input devices, and output loads. The example wiring diagram below on the right shows how this can work, but also the auxiliary supply output is an unused resource. You will want to avoid this situation if possible.



Sinking / Sourcing Concepts

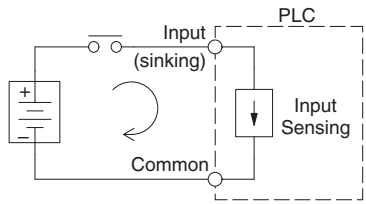
Before going further in the study of wiring strategies, you must have a solid understanding of "sinking" and "sourcing" concepts. Use of these terms occurs frequently in input or output circuit discussions. It is the goal of this section to make these concepts easy to understand, further ensuring your success in installation. First the following short definitions are provided, followed by practical applications.

Sinking = provides a path to supply ground (–)

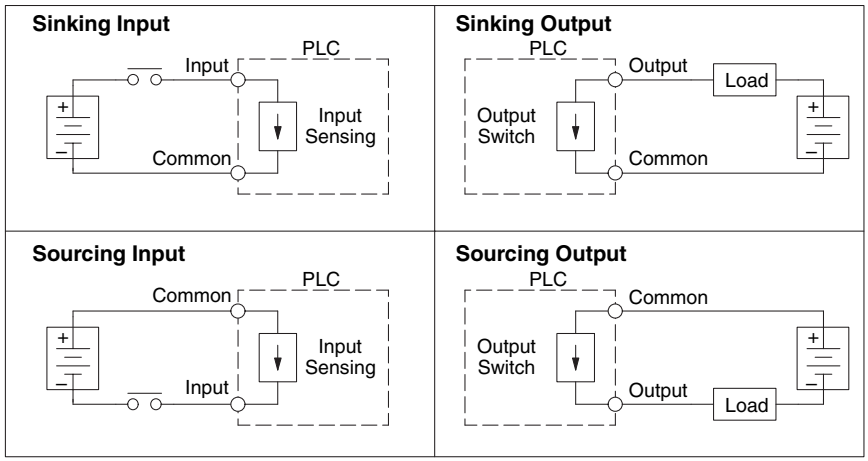
Sourcing = provides a path to supply source (+)

First you will notice these are only associated with DC circuits and not AC, because of the reference to (+) and (–) polarities. Therefore, sinking and sourcing terminology only applies to DC input and output circuits. Input and output points that are sinking only or sourcing only can conduct current in only one direction. This means it is possible to connect the external supply and field device to the I/O point with current trying to flow in the wrong direction, and the circuit will not operate. However, you can successfully connect the supply and field device every time by understanding "sourcing" and "sinking".

For example, the figure to the right depicts a "sinking" input. To properly connect the external supply, you will have to connect it so the input provides a path to ground (–). Start at the PLC input terminal, follow through the input sensing circuit, exit at the common terminal, and connect the supply (–) to the common terminal. By adding the switch, between the supply (+) and the input, the circuit has been completed . Current flows in the direction of the arrow when the switch is closed.

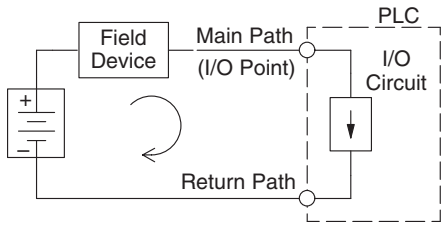


Apply the circuit principle above to the four possible combinations of input/output sinking/sourcing types as shown below. The I/O module specifications at the end of this chapter list the input or output type.

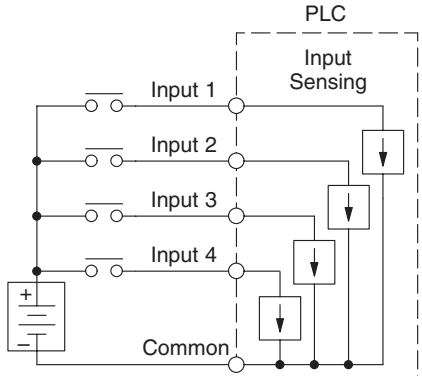


I/O "Common" Terminal Concepts

In order for a PLC I/O circuit to operate, current must enter at one terminal and exit at another. Therefore, at least two terminals are associated with every I/O point. In the figure to the right, the Input or Output terminal is the main path for the current. One additional terminal must provide the return path to the power supply.



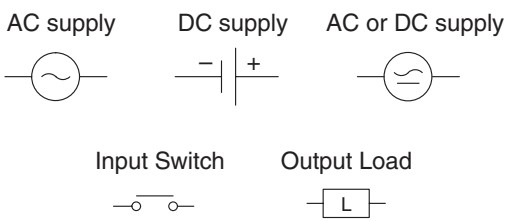
If there was unlimited space and budget for I/O terminals, every I/O point could have two dedicated terminals as the figure above shows. However, providing this level of flexibility is not practical or even necessary for most applications. So, most Input or Output points on PLCs are in groups which share the return path (called commons). The figure to the right shows a group (or bank) of four input points which share a common return path. In this way, the four inputs require only five terminals instead of eight.

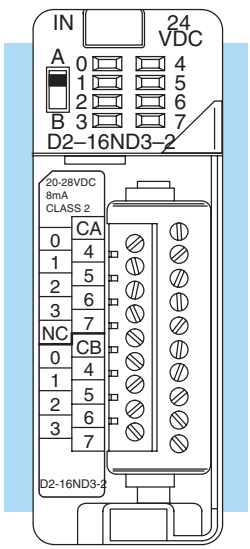


NOTE: *In the circuit above, the current in the common path is 4 times any channel's input current when all inputs are energized. This is especially important in output circuits, where heavier gauge wire is sometimes necessary on commons.*

Most DL205 input and output modules group their I/O points into banks that share a common return path. The best indication of I/O common grouping is on the wiring label, such as the one shown to the right. There are two circuit banks with eight input points in each. The common terminal for each is labeled "CA" and "CB", respectively.

In the wiring label example, the positive terminal of a DC supply connects to the common terminals. Some symbols you will see on the wiring labels, and their meanings are:



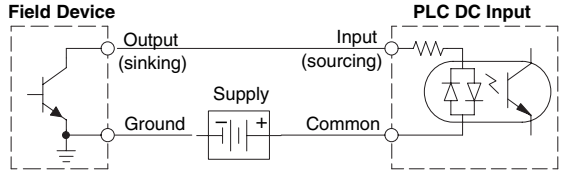


Connecting DC I/O to "Solid State" Field Devices

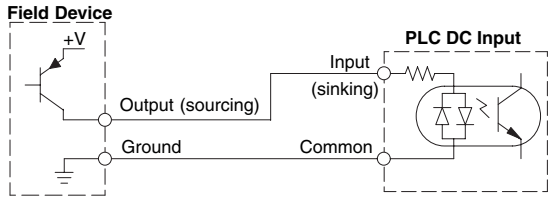
In the previous section on Sourcing and Sinking concepts, the DC I/O circuits were explained to sometimes only allow current to flow one way. This is also true for many of the field devices which have solid-state (transistor) interfaces. In other words, field devices can also be sourcing or sinking. *When connecting two devices in a series DC circuit, one must be wired as sourcing and the other as sinking.*

Solid State Input Sensors

Several DL205 DC input modules are flexible because they detect current flow in either direction, so they can be wired as either sourcing or sinking. In the following circuit, a field device has an open-collector NPN transistor output. It sinks current from the PLC input point, which sources current. The power supply can be the +24 auxiliary supply or another supply (+12VDC or +24VDC), as long as the input specifications are met.



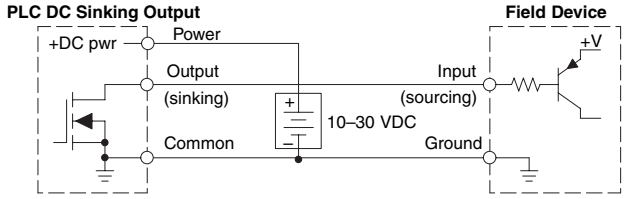
In the next circuit, a field device has an open-collector PNP transistor output. It sources current to the PLC input point, which sinks the current back to ground. Since the field device is sourcing current, no additional power supply is required.



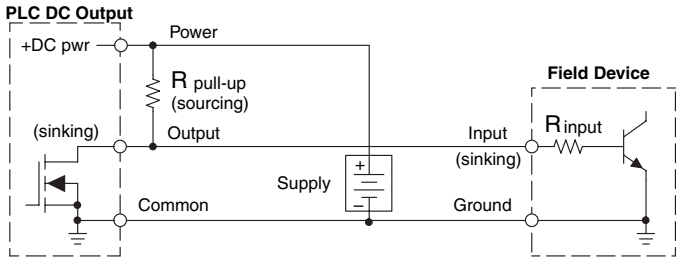
Solid State Output Loads

Sometimes an application requires connecting a PLC output point to a solid state input on a device. This type of connection is usually made to carry a low-level control signal, not to send DC power to an actuator.

Several of the DL205 DC output modules are the sinking type. This means that each DC output provides a path to ground when it is energized. In the following circuit, the PLC output point sinks current to the output common when energized. It is connected to a sourcing input of a field device input.



In the next example a PLC sinking DC output point is connected to the sinking input of a field device. This is a little tricky, because both the PLC output and field device input are sinking type. Since the circuit must have one sourcing and one sinking device, a sourcing capability needs to be added to the PLC output by using a pull-up resistor. In the circuit below, a $R_{pull\text{-}up}$ is connected from the output to the DC output circuit power input.



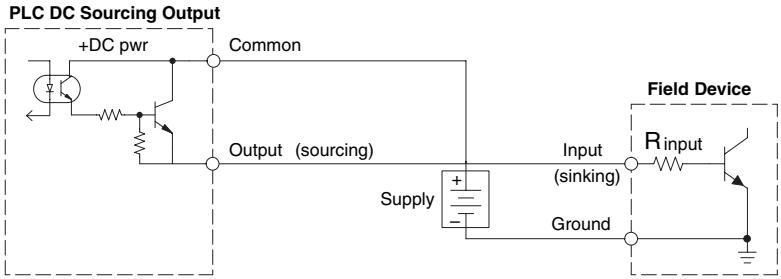
NOTE 1: *DO NOT attempt to drive a heavy load (>25mA) with this pull-up method*
NOTE 2: *Using the pull-up resistor to implement a sourcing output has the effect of inverting the output point logic. In other words, the field device input is energized when the PLC output is OFF, from a ladder logic point of view. Your ladder program must comprehend this and generate an inverted output. Or, you may choose to cancel the effect of the inversion elsewhere, such as in the field device.*

It is important to choose the correct value of $R_{pull\text{-}up}$. In order to do so, you need to know the nominal input current to the field device (I_{input}) when the input is energized. If this value is not known, it can be calculated as shown (a typical value is 15mA). Then use I_{input} and the voltage of the external supply to compute $R_{pull\text{-}up}$. Then calculate the power $P_{pull\text{-}up}$ (in watts), in order to size $R_{pull\text{-}up}$ properly.

$$I_{input} = \frac{V_{input\ (turn\text{–}on)}}{R_{input}}$$

$$R_{pull\text{-}up} = \frac{V_{supply} - 0.7}{I_{input}} - R_{input} \qquad P_{pull\text{-}up} = \frac{V_{supply}^2}{R_{pullup}}$$

Of course, the easiest way to drive a sinking input field device as shown below is to use a DC sourcing output module. The Darlington NPN stage will have about 1.5 V ON-state saturation, but this is not a problem with low-current solid-state loads.



Relay Output Guidelines

Several output modules in the DL205 I/O family feature relay outputs: D2–04TRS, D2–08TR, D2–12TR, D2–08CDR, F2–08TR and F2–08TRS. Relays are best for the following applications:

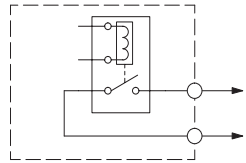
- Loads that require higher currents than the solid-state outputs can deliver
- Cost-sensitive applications
- Some output channels need isolation from other outputs (such as when some loads require different voltages than other loads)

Some applications in which NOT to use relays:

- Loads that require currents under 10mA
- Loads which must be switched at high speed or heavy duty cycle

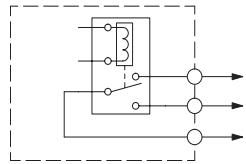
Relay outputs in the DL205 output modules are available in two contact arrangements, shown to the right. The Form A type, or SPST (single pole, single throw) type is normally open and is the simplest to use. The Form C type, or SPDT (single pole, double throw) type has a center contact which moves and a stationary contact on either side. This provides a normally closed contact and a normally open contact.

Relay with Form A contacts



Some relay output module's relays share common terminals, which connect to the wiper contact in each relay of the bank. Other relay modules have relays which are completely isolated from each other. In all cases, the module drives the relay coil when the corresponding output point is on.

Relay with Form C contacts



Relay Outputs – Transient Suppression for Inductive Loads in a Control System

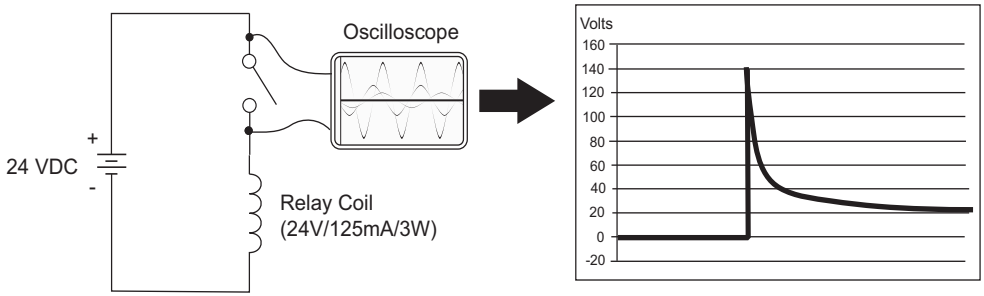
The following pages are intended to give a quick overview of the negative effects of transient voltages on a control system and provide some simple advice on how to effectively minimize them. The need for transient suppression is often not apparent to the newcomers in the automation world. Many mysterious errors that can afflict an installation can be traced back to a lack of transient suppression.

What is a Transient Voltage and Why is it Bad?

Inductive loads (devices with a coil) generate transient voltages as they transition from being energized to being de-energized. If not suppressed, the transient can be many times greater than the voltage applied to the coil. These transient voltages can damage PLC outputs or other electronic devices connected to the circuit, and cause unreliable operation of other electronics in the general area. Transients must be managed with suppressors for long component life and reliable operation of the control system.

This example shows a simple circuit with a small 24V/125mA/3W relay. As you can see, when the switch is opened, thereby de-energizing the coil, the transient voltage generated across the switch contacts peaks at 140V.

Example: Circuit with no Suppression



In the same circuit, replacing the relay with a larger 24V/290mA/7W relay will generate a transient voltage exceeding 800V (not shown). Transient voltages like this can cause many problems, including:

- Relay contacts driving the coil may experience arcing, which can pit the contacts and reduce the relay's lifespan.
- Solid state (transistor) outputs driving the coil can be damaged if the transient voltage exceeds the transistor's ratings. In extreme cases, complete failure of the output can occur the very first time a coil is de-energized.
- Input circuits, which might be connected to monitor the coil or the output driver, can also be damaged by the transient voltage.

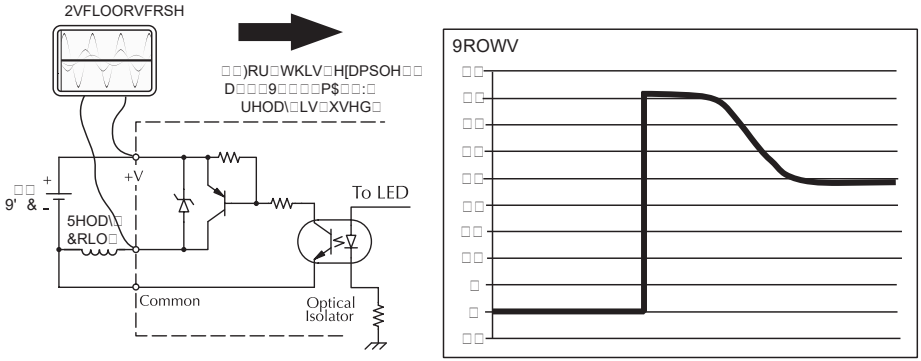
A very destructive side-effect of the arcing across relay contacts is the electromagnetic interference (EMI) it can cause. This occurs because the arcing causes a current surge, which releases RF energy. The entire length of wire between the relay contacts, the coil, and the power source carries the current surge and becomes an antenna that radiates the RF energy. It will readily couple into parallel wiring and may disrupt the PLC and other electronics in the area. This EMI can make an otherwise stable control system behave unpredictably at times.

PLC's Integrated Transient Suppressors

Although the PLC's outputs typically have integrated suppressors to protect against transients, they are not capable of handling them all. It is usually necessary to have some additional transient suppression for an inductive load.

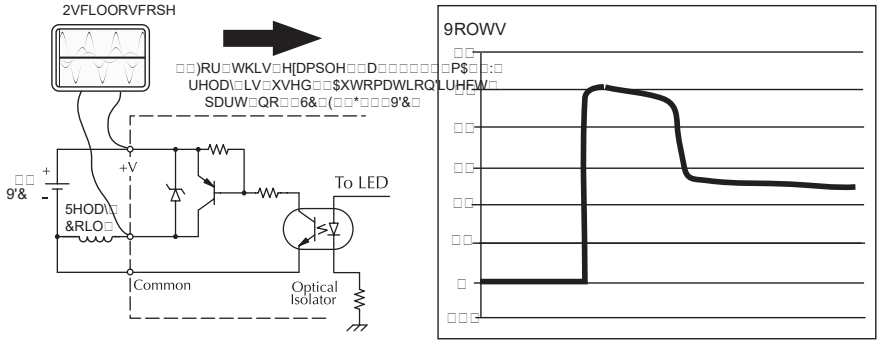
The next example uses the same 24V/125mA/3W relay used earlier. This example measures the PNP transistor output of a D0-06DD2 PLC, which incorporates an integrated Zener diode for transient suppression. Instead of the 140V peak in the first example, the transient voltage here is limited to about 40V by the Zener diode. While the PLC will probably tolerate repeated transients in this range for some time, the 40V is still beyond the module's peak output voltage rating of 30V.

Example: Small Inductive Load with Only Integrated Suppression



The next example uses the same circuit as above, but with a larger 24V/290mA/7W relay, thereby creating a larger inductive load. As you can see, the transient voltage generated is much worse, peaking at over 50V. Driving an inductive load of this size without additional transient suppression is very likely to permanently damage the PLC output.

Example: Larger Inductive Load with Only Integrated Suppression

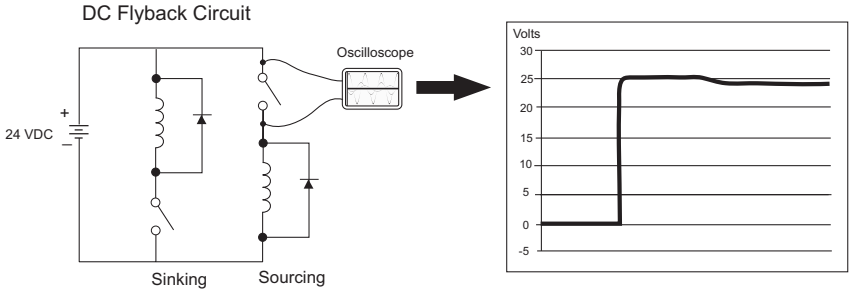


Additional transient suppression should be used in both these examples. If you are unable to measure the transients generated by the connected loads of your control system, using additional transient suppression on all inductive loads would be the safest practice.

Types of Additional Transient Protection

DC Coils:

The most effective protection against transients from a DC coil is a flyback diode. A flyback diode can reduce the transient to roughly 1V over the supply voltage, as shown in this example.



Many AutomationDirect socketed relays and motor starters have add-on flyback diodes that plug or screw into the base, such as the AD-ASMD-250 protection diode module and 784-4C-SKT-1 socket module shown below. If an add-on flyback diode is not available for your inductive load, an easy way to add one is to use AutomationDirect's DN-D10DR-A diode terminal block, a 600VDC power diode mounted in a slim DIN rail housing.

AD-ASMD-250
Protection Diode Module

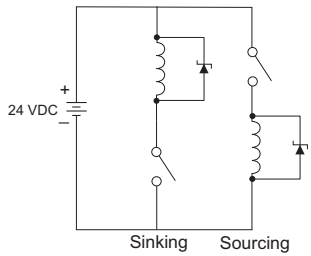
784-4C-SKT-1
Relay Socket

DN-D10DR-A
Diode Terminal Block

Two more common options for DC coils are Metal Oxide Varistors (MOV) or TVS diodes. These devices should be connected across the driver (PLC output) for best protection as shown below. The optimum voltage rating for the suppressor is the lowest rated voltage available that will NOT conduct at the supply voltage, while allowing a safe margin.

AutomationDirect's ZL-TSD8-24 transorb module is a good choice for 24VDC circuits. It is a bank of 8 uni-directional 30V TVS diodes. Since they are uni-directional, be sure to observe the polarity during installation. MOVs or bi-directional TVS diodes would install at the same location, but have no polarity concerns.

DC MOV or TVS Diode Circuit

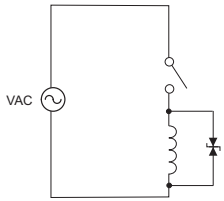


AC Coils:

Two options for AC coils are MOVs or bi-directional TVS diodes. These devices are most effective at protecting the driver from a transient voltage when connected across the driver (PLC output) but are also commonly connected across the coil. The optimum voltage rating for the suppressor is the lowest rated voltage available that will NOT conduct at the supply voltage, while allowing a safe margin.

AutomationDirect's ZL-TSD8-120 transorb module is a good choice for 120VAC circuits. It is a bank of eight bi-directional 180V TVS diodes.

AC MOV or Bi-Directional Diode Circuit



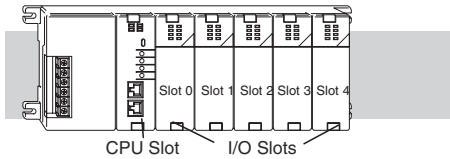
NOTE: *Manufacturers of devices with coils frequently offer MOV or TVS diode suppressors as an add-on option which mount conveniently across the coil. Before using them, carefully check the suppressor's ratings. Just because the suppressor is made specifically for that part does not mean it will reduce the transient voltages to an acceptable level.*

For example, a MOV or TVS diode rated for use on 24-48 VDC coils would need to have a high enough voltage rating to NOT conduct at 48V. That suppressor might typically start conducting at roughly 60VDC. If it were mounted across a 24V coil, transients of roughly 84V (if sinking output) or -60V (if sourcing output) could reach the PLC output. Many semiconductor PLC outputs cannot tolerate such levels.

I/O Module Positioning, Wiring, and Specification

Slot Numbering

The DL205 bases each provide different numbers of slots for use with the I/O modules. You may notice the bases refer to 3-slot, 4-slot, etc. One of the slots is dedicated to the CPU, so you always have one less I/O slot. For example, you have five I/O slots with a 6-slot base. The I/O slots are numbered 0–4. The CPU slot always contains a PLC CPU or other CPU–slot controller and is not numbered.



Module Placement Restrictions

The following table lists the valid locations for all types of modules in a DL205 system:

Module/Unit	Local CPU Base	Local Expansion Base	Remote I/O Base
CPUs	CPU Slot Only		
DC Input Modules .	√	√	√
AC Input Modules	√	√	√
DC Output Modules	√	√	√
AC Output Modules	√	√	√
Relay Output Modules	√	√	√
Analog Input and Output Modules	√	√	√
Local Expansion			
Base Expansion Module	√	√	
Base Controller Module		CPU Slot Only	
Serial Remote I/O			
Remote Master	√		
Remote Slave Unit			CPU Slot Only
Ethernet Remote Master	√		
CPU Interface			
Ethernet Base Controller	Slot 0 Only		Slot 0 Only*
WinPLC	Slot 0 Only		
DeviceNet	Slot 0 Only		
Profibus	Slot 0 Only		
SDS	Slot 0 Only		
Specialty Modules			
Counter Interface **	Slot 0 Only		
Counter I/O	√		√*
Data Communications	√		
Ethernet Communications	√		
BASIC CoProcessor	√		
Simulator	√	√	√
Filler	√	√	√

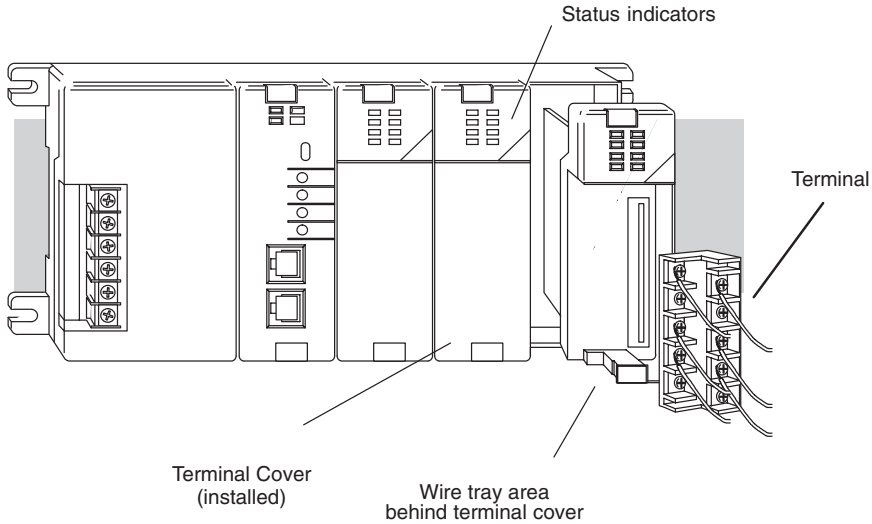
* When used with H2-ERM(100) Ethernet Remote I/O system. ** D2-262 does not support D2-CTRINT module.

Special Placement Considerations for Analog Modules

In most cases, the analog modules can be placed in any slot. However, the placement can also depend on the type of CPU you are using and the other types of modules installed to the left of the analog modules. If you're using a D2-230 CPU (or a D2-240 CPU with firmware earlier than V1.4) you should check the DL205 Analog I/O Manual for any possible placement restrictions related to your particular module. You can download the DL205 Analog I/O Manual (part number D2–ANLG–M) from **automationdirect.com**.

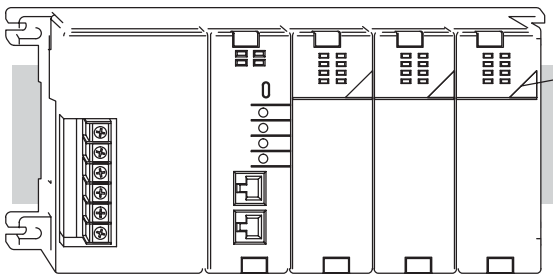
Discrete Input Module Status Indicators

The discrete modules provide LED status indicators to show the status of the input points.



Color Coding of I/O Modules

The DL205 family of I/O modules have a color coding scheme to help you quickly identify if a module is either an input module, output module, or a specialty module. This is done through a color bar indicator located on the front of each module. The color scheme is listed below:



Module Type	Color Code
Discrete/Analog Output	Red
Discrete/Analog Input	Blue
Other	White

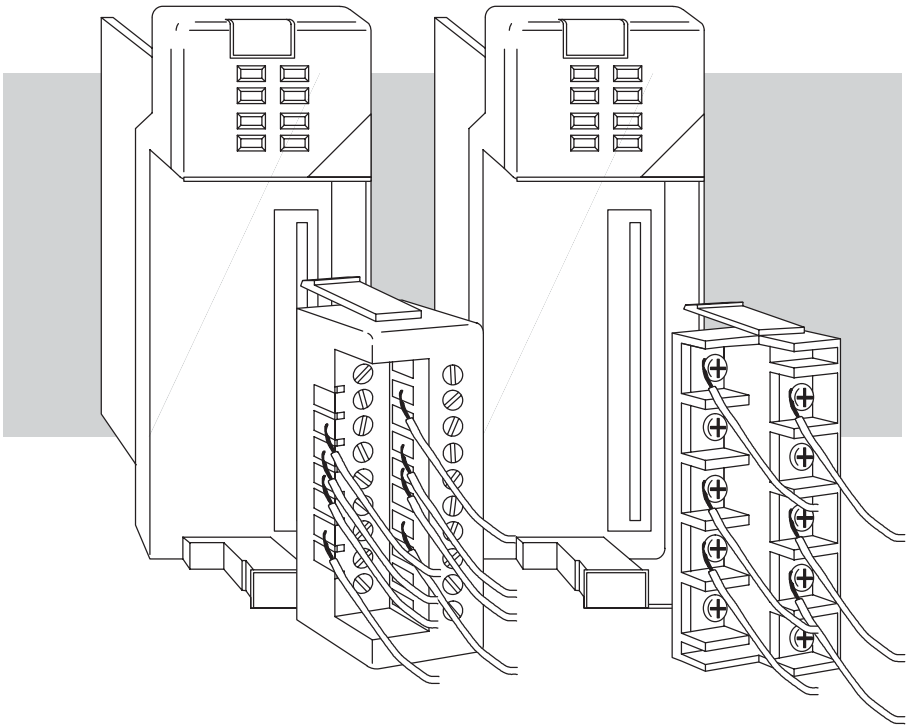
Wiring the Different Module Connectors

There are two types of module connectors for the DL205 I/O. Some modules have normal screw terminal connectors. Other modules have connectors with recessed screws. The recessed screws help minimize the risk of someone accidentally touching active wiring.

Both types of connectors can be easily removed. If you examine the connectors closely, you'll notice there are squeeze tabs on the top and bottom. To remove the terminal block, press the squeeze tabs and pull the terminal block away from the module.

We also have DIN rail mounted terminal blocks, DINnectors (refer to our catalog for a complete listing of all available products). *ZIP*Links come with special pre–assembled cables with the I/O connectors installed and wired.

WARNING: For some modules, field device power may still be present on the terminal block even though the PLC system is turned off. To minimize the risk of electrical shock, check all field device power before you remove the connector.



I/O Wiring Checklist

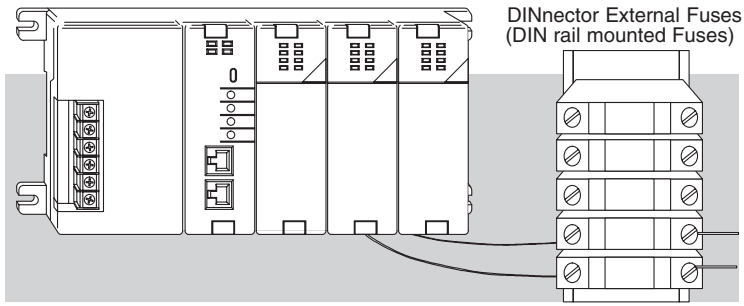
Use the following guidelines when wiring the I/O modules in your system.

1. There is a limit to the size of wire the modules can accept. The table below lists the suggested AWG for each module type. When making terminal connections, follow the suggested torque values.

Terminal type	Suggested AWG Range	Suggested Torque
10-Terminal Fixed	14 – 24 AWG	3.5 lb-inch (0.4 N·m)
10-Terminal Removable	16* – 24 AWG	7.81 lb-inch (0.88 N·m)
20-Terminal Removable	16* – 24 AWG	2.65 lb-in (0.3 N·m)

NOTE: 16 AWG Type TFFN or Type MTW is recommended.** Other types of 16 AWG may be acceptable, but it really depends on the thickness and stiffness of the wire insulation. **If the insulation is too thick or stiff and a majority of the module's I/O points are used, then the plastic terminal cover may not close properly or the connector may pull away from the module. This applies especially for high temperature thermoplastics such as THHN.

2. Always use a continuous length of wire, do not combine wires to attain a needed length.
3. Use the shortest possible wire length.
4. Use wire trays for routing where possible.
5. Avoid running wires near high energy wiring. Also, avoid running input wiring close to output wiring where possible.
6. To minimize voltage drops when wires must run a long distance, consider using multiple wires for the return line.
7. Avoid running DC wiring in close proximity to AC wiring where possible.
8. Avoid creating sharp bends in the wires.
9. To reduce the risk of having a module with a blown fuse, we suggest you add external fuses to your I/O wiring. A fast blow fuse, with a lower current rating than the I/O module fuse can be added to each common, or a fuse with a rating of slightly less than the maximum current per output point can be added to each output. Refer to our catalog for a complete line of DINnectors, DIN-rail mounted fuse blocks.



DL205 I/O Module Specifications

The tables below list the Input and Output modules for the DL205 PLC system. Specifications for each module begin on the following page.

DL205 Input Module Chart

Modules Types	Number of Input Points	DC Current Sink Input	DC Current Source Input	AC Input
D2-08ND3	8	✓	✓	–
D2-16ND3-2	16	✓	✓	–
D2-32ND3	32	✓	✓	–
D2-32ND3-2	32	✓	✓	–
D2-08NA-1	8	–	–	✓
D2-08NA-2	8	–	–	✓
D2-16NA	16	–	–	✓
F2-08SIM	8	–	–	–
*D2-08CDR**	4	✓	✓	–

* D2-08CDR is a combo module, 4 pt. DC Input and 4 pt. Relay Output

DL205 Output Module Chart

Modules Types	Number of Output Points	DC Current Sink Output	DC Current Source Output	AC Output
D2-04TD1	4	✓	–	–
D2-08TD1	8	✓	–	–
D2-08TD2	8	–	✓	–
D2-16TD1-2	16	✓	–	–
D2-16TD2-2	16	–	✓	–
F2-16TD1P	16	✓	–	–
F2-16TD2P	16	–	✓	–
D2-32TD1	32	✓	–	–
D2-32TD2	32	–	✓	–
F2-08TA	8	–	–	✓
D2-08TA	8	–	–	✓
D2-12TA	12	–	–	✓
D2-04TRS	4	✓	✓	✓
D2-08TR	8	✓	✓	✓
F2-08TR	8	✓	✓	✓
F2-08TRS	8	✓	✓	✓
D2-12TR	12	✓	✓	✓
*D2-08CDR**	4	✓	✓	✓

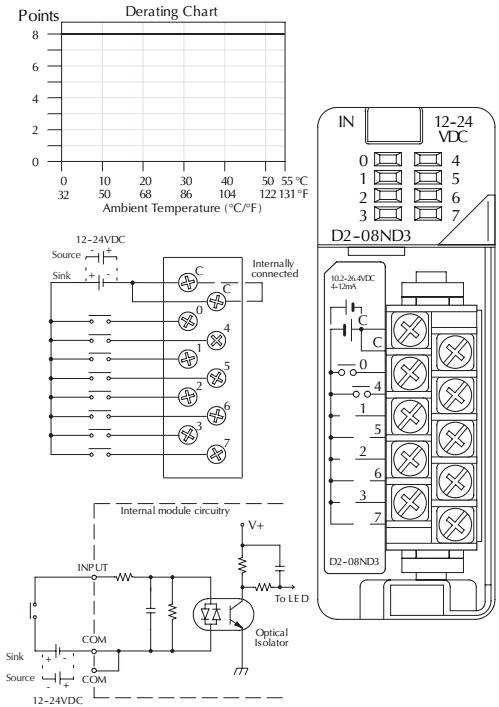
* D2-08CDR is a combo module, 4 pt. DC Input and 4 pt. Relay Output

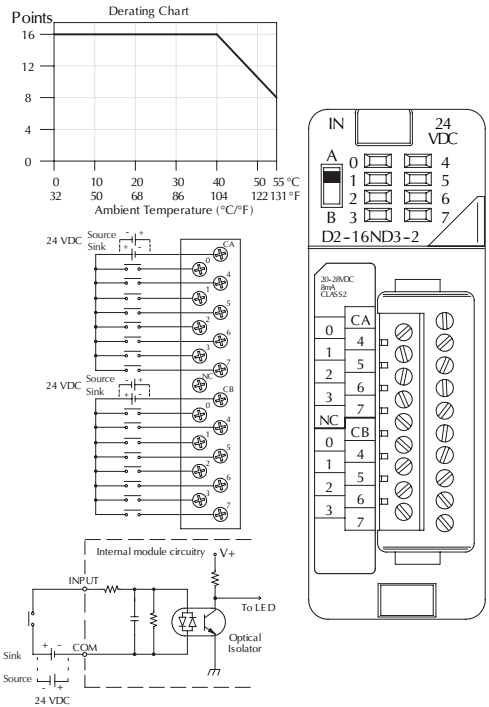
D2-08ND3, DC Input

D2-08ND3 DC Input	
Inputs per Module	8 (sink/source)
Commons per Module	1 (2 I/O terminal points)
Input Voltage Range	10.2–26.4 VDC
Peak Voltage	26.4 VDC
ON Voltage Level	9.5 VDC minimum
OFF Voltage Level	3.5 VDC maximum
AC Frequency	N/A
Input Impedance	2.7 kΩ
Input Current	4.0 mA @ 12VDC 8.5 mA @ 24VDC
Minimum ON Current	3.5 mA
Maximum OFF Current	1.5 mA
Base Power Required 5VDC	50mA
OFF to ON Response	1 to 8 ms
ON to OFF Response	1 to 8 ms
Terminal Type (included)	Removable, D2-8IOCON
Status Indicator	Logic side
Weight	2.3 oz. (65g)

D2-16ND3-2, DC Input

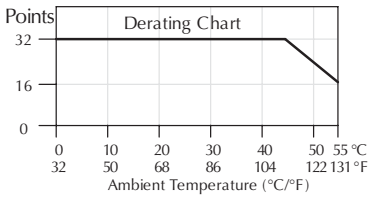
D2-16ND3-2 DC Input	
Inputs per Module	16 (sink/source)
Commons per Module	2 isolated (8 I/O terminal points/com)
Input Voltage Range	20–28 VDC
Peak Voltage	30VDC (10mA)
ON Voltage Level	19 VDC minimum
OFF Voltage Level	7VDC maximum
AC Frequency	N/A
Input Impedance	3.9 kΩ
Input Current	6mA @ 24VDC
Minimum ON Current	3.5 mA
Maximum OFF Current	1.5 mA
Base Power Required 5VDC	100mA
OFF to ON Response	3 to 9 ms
ON to OFF Response	3 to 9 ms
Terminal Type (included)	Removable, D2-16IOCON
Status Indicator	Logic side
Weight	2.3 oz. (65g)

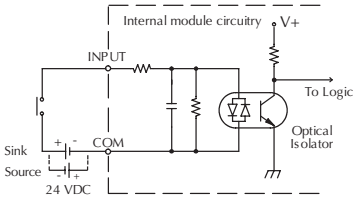


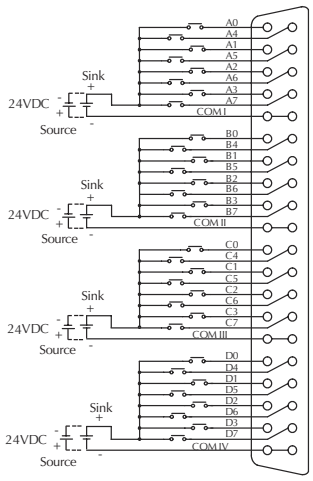


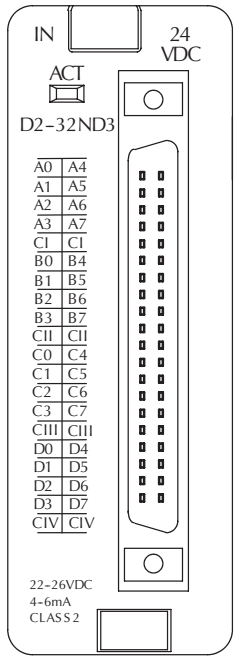
D2–32ND3, DC Input

D2-32ND3 DC Input	
Inputs per Module	32 (sink/source)
Commons per Module	4 isolated (8 I/O terminal points / com)
Input Voltage Range	20-28 VDC
Peak Voltage	30VDC
ON Voltage Level	19VDC minimum
OFF Voltage Level	7VDC maximum
AC Frequency	N/A
Input Impedance	4.8 kΩ
Input Current	8.0 mA @ 24 VDC
Minimum ON Current	3.5 mA
Maximum OFF Current	1.5 mA
Base Power Required 5VDC	25mA
OFF to ON Response	3 to 9 ms
ON to OFF Response	3 to 9 ms
Terminal Type (not included)	40-pin Connector[1]
Status Indicator	Module Activity LED
Weight	2.1 oz. (60 g)
[1] Connector sold separately. See Terminal Blocks and Wiring for wiring options.	





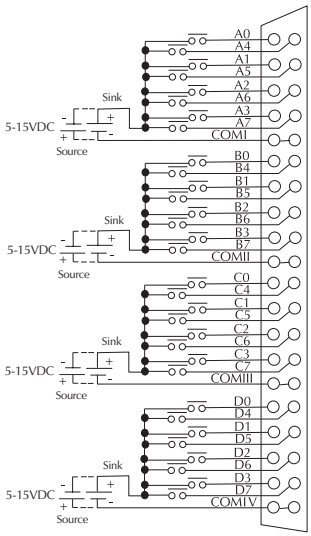


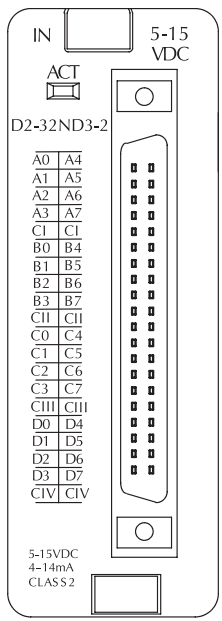


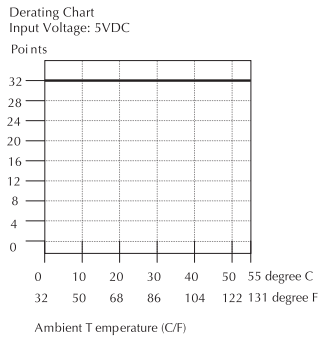
D2–32ND3–2, DC Input

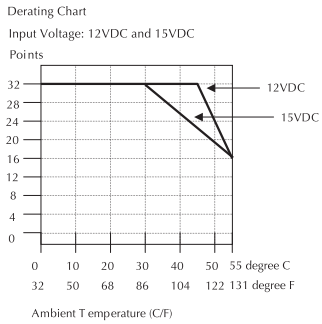
D2-32ND3-2 DC Input	
Inputs per Module	32 (Sink/Source)
Commons per Module	4 isolated (8 I/O terminal points / com)
Input Voltage Range	4.50 to 15.6 VDC min. to max.
Peak Voltage	16VDC
ON Voltage Level	4VDC minimum
OFF Voltage Level	2VDC maximum
AC Frequency	N/A
Input Impedance	1.0 kΩ @ 5–5 VDC
Input Current	4mA @ 5VDC 11mA @ 12VDC 14mA @ 15VDC
Maximum Input Current	16mA @ 15.6 VDC
Minimum ON Current	3mA
Maximum OFF Current	0.5 mA
Base Power Required 5VDC	25mA
OFF to ON Response	3 to 9 ms
ON to OFF Response	3 to 9 ms
Terminal Type (not included)	40-pin connector[1]
Status Indicator	Module activity LED
Weight	2.1 oz. (60g)

[1] Connector sold separately.
See Terminal Blocks and Wiring for wiring options.



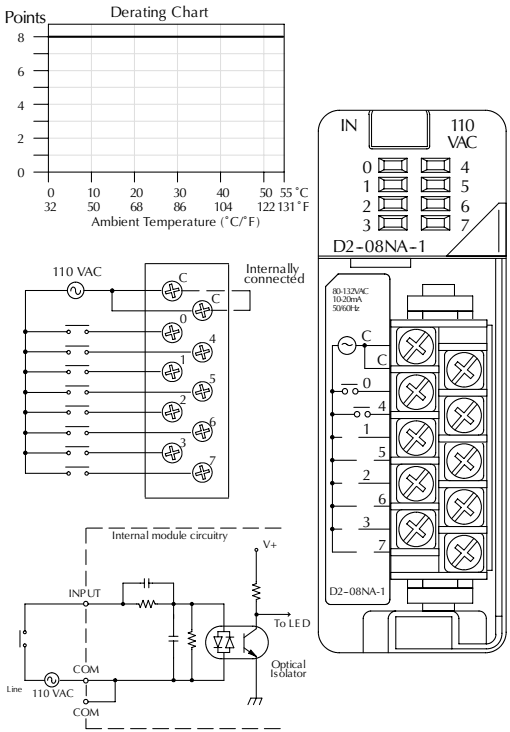






D2-08NA-1, AC Input

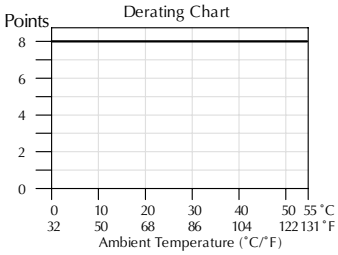
D2-08NA-1 AC Input	
Inputs per Module	8
Commons per Module	1 (2 I/O terminal points)
Input Voltage Range	80–132 VAC
Peak Voltage	132VAC
ON Voltage Level	75VAC minimum
OFF Voltage Level	20VAC maximum
AC Frequency	47–63 Hz
Input Impedance	12kΩ @ 60Hz
Input Current	13mA @ 100VAC, 60Hz 11mA @ 100 VAC, 50Hz
Minimum ON Current	5mA
Maximum OFF Current	2mA
Base Power Required 5VDC	50mA
OFF to ON Response	5 to 30 ms
ON to OFF Response	10 to 50 ms
Terminal Type (included)	Removable; D2-8IOCON
Status Indicator	Logic side
Weight	2.5 oz. (70g)

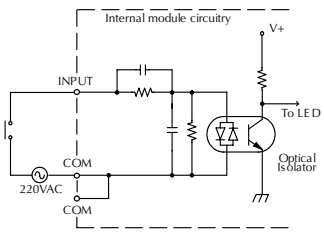


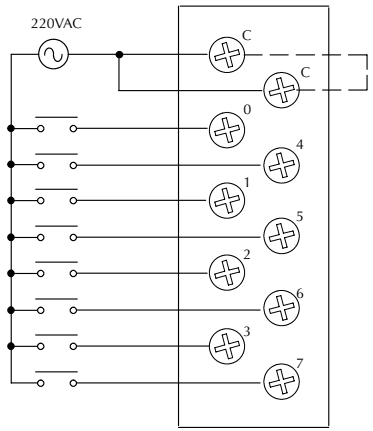
D2-08NA-2, AC Input

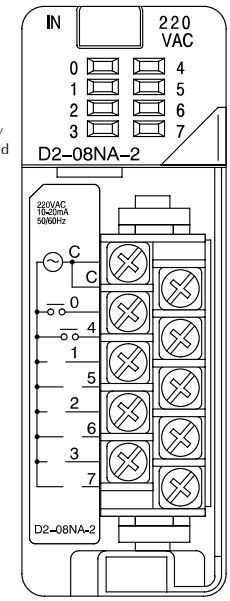
D2-08NA-2	AC Input
Inputs per Module	8
Commons per Module	1 (2 I/O terminal points)
Input Voltage Range	170–265 VAC
Peak Voltage	265VAC
ON Voltage Level	150VAC minimum
OFF Voltage Level	40VAC maximum
AC Frequency	47–63 Hz
Input Impedance	18kΩ @ 60Hz
Input Current	9mA @ 220VAC, 50Hz 11mA @ 265VAC, 50Hz 10mA @ 220VAC, 60Hz 12mA @ 265VAC, 60Hz
Minimum ON Current	10mA
Maximum OFF Current	2mA
Base Power Required 5VDC	100mA
OFF to ON Response	5 to 30 ms
ON to OFF Response	10 to 50 ms
Terminal Type (included)	Removable; D2-8IOCON
Status Indicator	Logic side
Weight	2.5 oz. (70g)

Operating Temperature	32°F to 131°F (0° to 55°C)
Storage Temperature	-4°F to 158°F (-20°C to 70°C)
Humidity	35% to 95% (non-condensing)
Atmosphere	No corrosive gases permitted
Vibration	MIL STD 810C 514.2
Shock	MIL STD 810C 516.2
Insulation Withstand Voltage	1,500VAC 1 minute (COM-GND)
Insulation Resistance	10M ≃ @ 500VDC
Noise Immunity	NEMA 1,500V 1 minute SANKI 1,000V 1 minute
RFI	150MHz, 430MHz



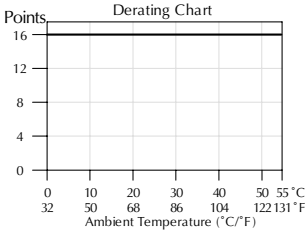


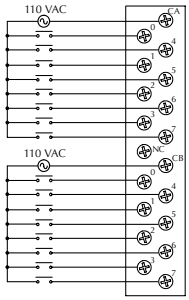


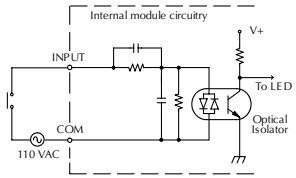


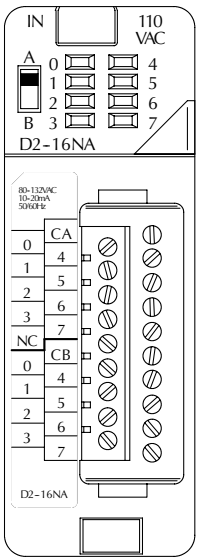
D2-16NA, AC Input

D2-16NA AC Input	
Inputs per Module	16
Commons per Module	2 (isolated)
Input Voltage Range	80–132 VAC
Peak Voltage	132VAC
ON Voltage Level	70VAC minimum
OFF Voltage Level	20VAC maximum
AC Frequency	47–63 Hz
Input Impedance	12kΩ @ 60Hz
Input Current	11mA @ 100VAC, 50Hz 13mA @ 100VAC, 60Hz 15mA @ 132VAC, 60Hz
Minimum ON Current	5mA
Maximum OFF Current	2mA
Base Power Required 5VDC	100mA
OFF to ON Response	5 to 30 ms
ON to OFF Response	10 to 50 ms
Terminal Type (included)	Removable; D2-16IOCON
Status Indicator	Logic side
Weight	2.4 oz. (68g)



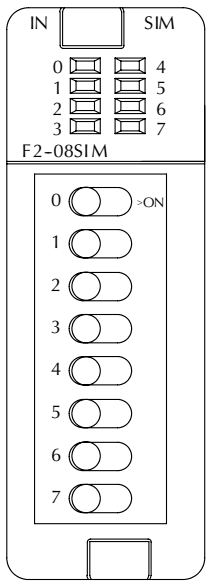






F2-08SIM, Input Simulator

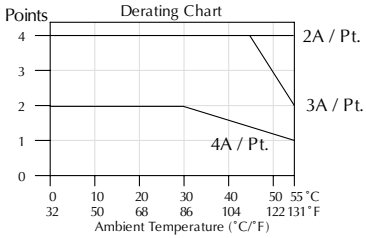
F2-08SIM Input Simulator	
Inputs per Module	8
Base Power Required 5VDC	50mA
Terminal Type	None
Status Indicator	Switch side
Weight	2.65 oz. (75g)

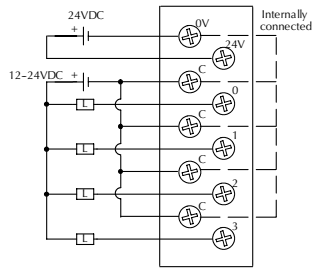


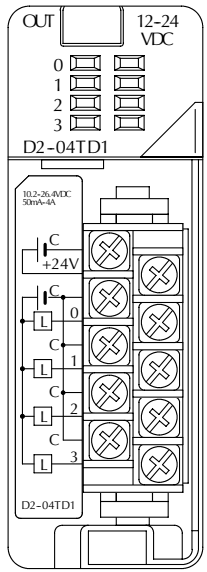
D2-04TD1, DC Output

D2-04TD1 DC Output	
Outputs per Module	4 (current sinking)
Output Points Consumed	8 points (only first 4 pts. used)
Commons per Module	1 (4 I/O terminal points)
Output Type	NMOS FET (open drain)
Operating Voltage	10.2-26.4 VDC
Peak Voltage	40VDC
ON Voltage Drop	0.72 VDC maximum
AC Frequency	N/A
Max Load Current (resistive)	4A/point 8A/common
Max Leakage Current	0.1 mA @ 40 VDC
Max Inrush Current	6A for 100 ms, 15A for 10ms
Minimum Load Current	50mA

External DC Required	24VDC @ 20mA max.
Base Power Required 5VDC	60mA
OFF to ON Response	1ms
ON to OFF Response	1ms
Terminal Type (included)	Removable; D2-8IOCON
Status Indicator	Logic side
Weight	2.8 oz. (80g)
Fuses	4 (1 per point) (6.3 A slow blow, non-replaceable)







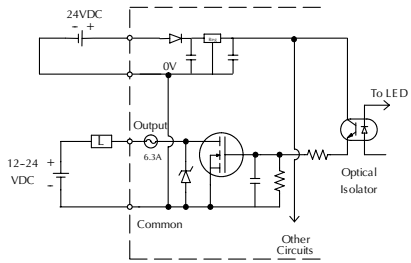
Inductive Load
Maximum Number of Switching Cycles per Minute

Load Current	Duration of output in ON state 7ms	40ms	100ms
0.1A	8000	1400	600
0.5A	1600	300	120
1.0A	800	140	60
1.5A	540	90	35
2.0A	400	70	-
3.0A	270	-	-
4.0A	200	-	-

At 40mS duration, loads of 3.0 A or greater cannot be used.

At 100mS duration, loads of 2.0 A or greater cannot be used.

Find the load current you expect to use and the duration that the output is ON. The number at the intersection of the row and column represents the switching cycles per minute. For example, a 1A inductive load that is on for 100ms can be switched on and off a maximum of 60 times per minute. To convert this to duty cycle percentage use: (duration x cycles)/60. In this example, (60 x .1)/60 = .1, or 10% duty cycle.

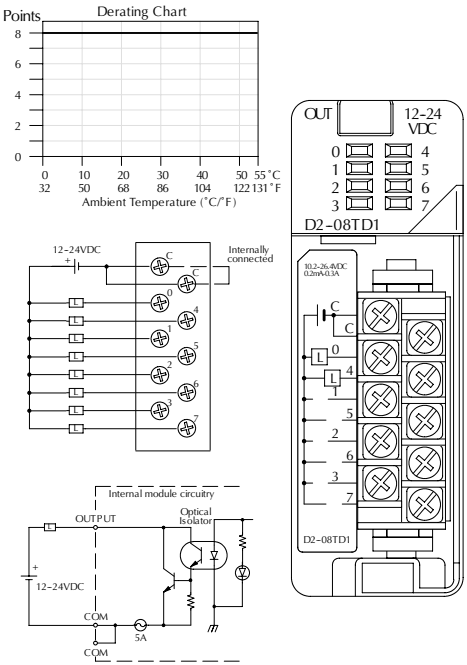


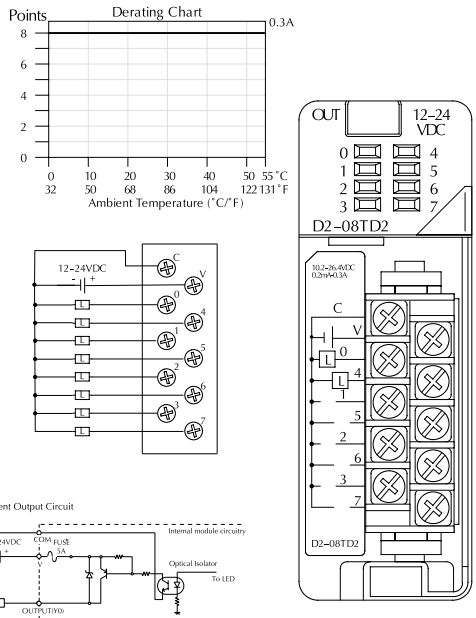
D2–08TD1, DC Output

D2-08TD1 DC Output	
Outputs per Module	8 (current sinking)
Commons per Module	1 (2 I/O terminal points)
Output Type	NPN open collector
Operating Voltage	10.2–26.4 VDC
Peak Voltage	40VDC
ON Voltage Drop	1.5 VDC maximum
AC Frequency	N/A
Minimum Load Current	0.5 mA
Max Load Current	0.3 A/point; 2.4 A/common
Max Leakage Current	0.1 mA @ 40VDC
Max Inrush Current	1A for 10ms
Base Power Required 5VDC	100mA
OFF to ON Response	1ms
ON to OFF Response	1ms
Terminal Type (included)	Removable; D2-8IOCON
Status Indicator	Logic side
Weight	2.3 oz. (65g)
Fuses	1 per common 5A fast blow, non-replaceable

D2–08TD2, DC Output

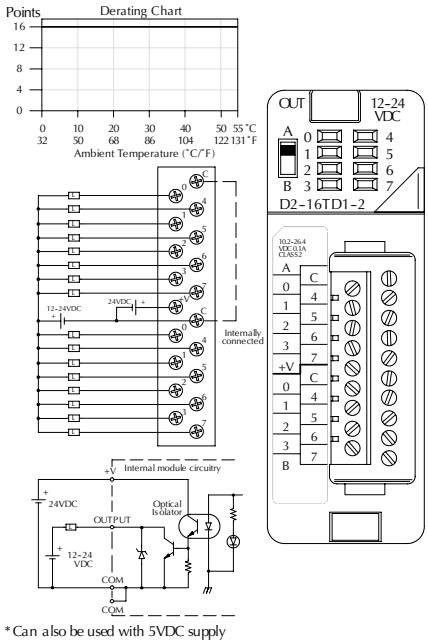
D2-08TD2 DC Output	
Outputs per Module	8 (current sourcing)
Commons per Module	1
Output Type	PNP open collector
Operating Voltage	12–24 VDC
Output Voltage	10.8–26.4 VDC
Peak Voltage	40VDC
ON Voltage Drop	1.5 VDC
AC Frequency	N/A
Minimum Load Current	N/A
Max Load Current	0.3 A per point; 2.4 A per common
Max Leakage Current	1.0 mA @ 40VDC
Max Inrush Current	1A for 10 ms
Base Power Required 5VDC	100 mA
OFF to ON Response	1ms
ON to OFF Response	1ms
Terminal Type (included)	Removable; D2-8IOCON
Status Indicator	Logic side
Weight	2.1 oz. (60g)
Fuse	5A fast blow, non-replaceable





D2–16TD1–2, DC Output

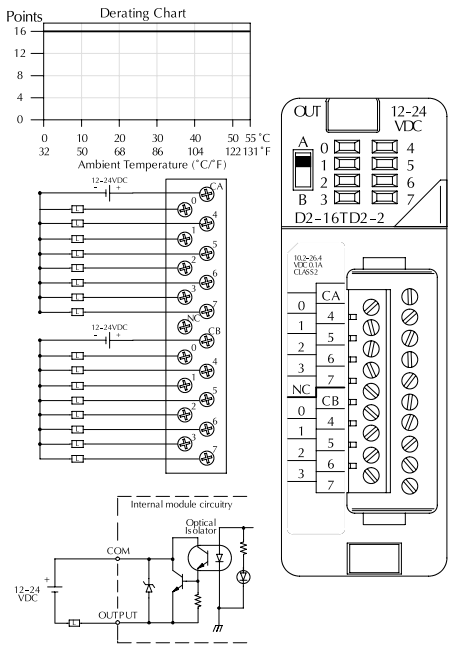
D2-16TD1-2 DC Output	
Outputs per Module	16 (current sinking)
Commons per Module	1 (2 I/O terminal points)
Output Type	NPN open collector
External DC required	24VDC ±4V @ 80mA max
Operating Voltage	10.2-26.4 VDC
Peak Voltage	30VDC
ON Voltage Drop	0.5 VDC maximum
AC Frequency	N/A
Minimum Load Current	0.2 mA
Max Load Current	0.1 A/point 1.6 A/common
Max Leakage Current	0.1 mA @ 30 VDC
Max Inrush Current	150mA for 10 ms
Base Power Required 5VDC	200mA
OFF to ON Response	0.5 ms
ON to OFF Response	0.5 ms
Terminal Type (included)	Removable; D2-16IOCON
Status Indicator	Logic side
Weight	2.3 oz. (65g)
Fuses	None



*Can also be used with 5VDC supply

D2–16TD2–2, DC Output

D2-16TD2-2 DC Output	
Outputs per Module	16 (current sourcing)
Commons per Module	2
Output Type	NPN open collector
Operating Voltage	10.2-26.4 VDC
Peak Voltage	30VDC
ON Voltage Drop	1.0 VDC maximum
AC Frequency	N/A
Minimum Load Current	0.2 mA
Max Load Current	0.1 A/point 1.6 A/ module
Max Leakage Current	0.1 mA @ 30 VDC
Max Inrush Current	150mA for 10 ms
Base Power Required 5VDC	200mA
OFF to ON Response	0.5 ms
ON to OFF Response	0.5 ms
Terminal Type (included)	Removable; D2-16IOCON
Status Indicator	Logic side
Weight	2.8 oz. (80g)
Fuses	None



F2–16TD1(2)P, DC Output With Fault Protection

NOTE: *Not supported in D2-230, D2-240 and D2-250 CPUs.*

These modules detect the following fault status and turn the related X bit(s) on.

1. Missing external 24VDC for the module
2. Open load[1]
3. Over temperature (the output is shut down)
4. Over load current (the output is shut down)

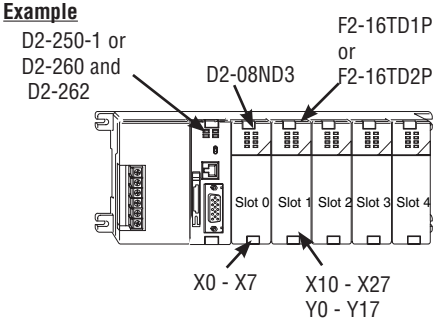
Fault Status	X bit Fault Status Indication
Missing external 24VDC	All 16 X bits are on.
Open load[1]	Only the X bit assigned to the faulted output is on
Over temperature	
Over load current	

When these modules are installed, 16 X bits are automatically assigned as the fault status indicator. Each X bit indicates the fault status of each output.

In this example, X10-X27 are assigned as the fault status indicator.

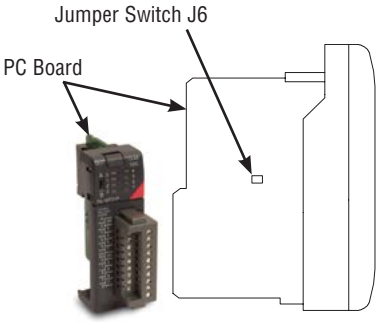
X10: Fault status indicator for Y0
X11: Fault status indicator for Y1
↓
X26: Fault status indicator for Y16
X27: Fault status indicator for Y17

Example



The fault status indicators (X bits) can be reset by performing the indicated operations in the following table:

Fault Status	Operation
Missing external 24VDC	Apply external 24VDC
Open load[1]	Connect the load.
Over temperature	Turn the output (Y bit) off or power cycle the PLC
Over load current	



NOTE 1*: Open load detection can be disabled by removing the jumper switch J6 on the module PC board.*

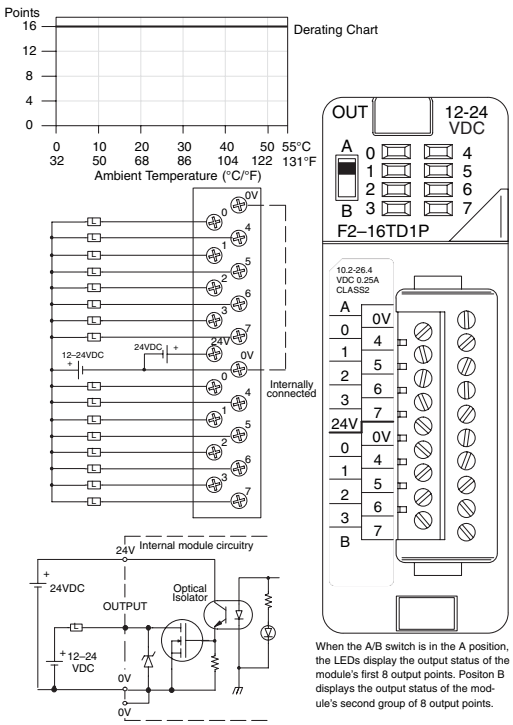
Continued on next two pages.

F2–16TD1P, DC Output With Fault Protection

***NOTE**: Not supported in D2-230, D2-240 and D2-250 CPUs.*

***NOTE**: Supporting Firmware:*
D2-250-1 must be v4.80 or later
D2-260 must be v2.60 or later
D2-262 must be v1.00 or later

F2-16TD1P DC Output with Fault Protection	
Inputs per module	16 (status indication)
Outputs per module	16 (current sinking)
Commons per module	1 (2 I/O terminal points)
Output type	NMOS FET (open drain)
Operating voltage	10.2–26.4 VDC, external
Peak voltage	40VDC
AC frequency	N/A
ON voltage drop	0.7 V (output current 0.5 A)
Overcurrent trip	0.6 A, min., 1.2A, max.
Minimum load current	0.2 mA
Maximum load current	0.25 A/point; 4A/common
Max leakage current	0.2 mA (load detect enabled); 0.3 mA disabled
Max inrush current	150mA for 10ms
Base power required 5V	70mA
OFF to ON response	0.5 ms
ON to OFF response	0.5 ms
Terminal type	Removable (D2-16IOCON)
Status indicators	Logic Side
Weight	2.0 oz. (25g)
Fuses	None
External DC required	24VDC ±10% @ 50mA
External DC overvoltage shutdown	27V, outputs are restored when voltage is within limits



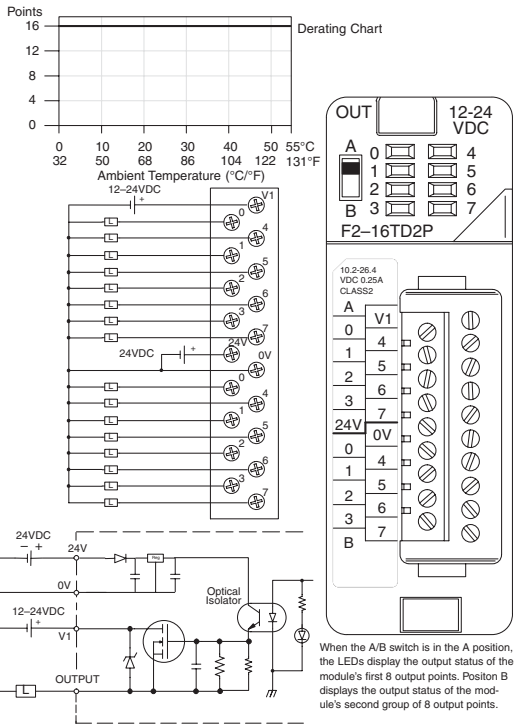
When the A/B switch is in the A position, the LEDs display the output status of the module's first 8 output points. Positon B displays the output status of the module's second group of 8 output points.

F2–16TD2P, DC Output with Fault Protection

***NOTE**: Not supported in D2-230, D2-240 and D2-250 CPUs.*

***NOTE**: Supporting Firmware:*
D2-250-1 must be v4.80 or later
D2-260 must be v2.60 or later
D2-262 must be v1.00 or later.

F2-16TD2P DC Output with Fault Protection	
Inputs per module	16 (status indication)
Outputs per module	16 (current sourcing)
Commons per module	1
Output type	NMOS FET (open source)
Operating voltage	10.2–26.4 VDC, external
Peak voltage	40VDC
AC frequency	N/A
ON voltage drop	0.7 V (output current 0.5 A)
Overcurrent trip	0.6 A min., 1.2 A max.
Minimum load current	0.2 mA
Maximum load current	0.25 A/point; 4A/common
Max leakage current	0.2 mA (load detect enabled); 0.3 mA disabled
Max inrush current	150mA for 10ms
Base power required 5V	70mA
OFF to ON response	0.5 ms
ON to OFF response	0.5 ms
Terminal type	Removable (D2-16IOCON)
Status indicators	Logic Side
Weight	2.0 oz. (25g)
Fuses	None
External DC required	24VDC ±10% @ 50mA
External DC overvoltage shutdown	27V, outputs are restored when voltage is within limits

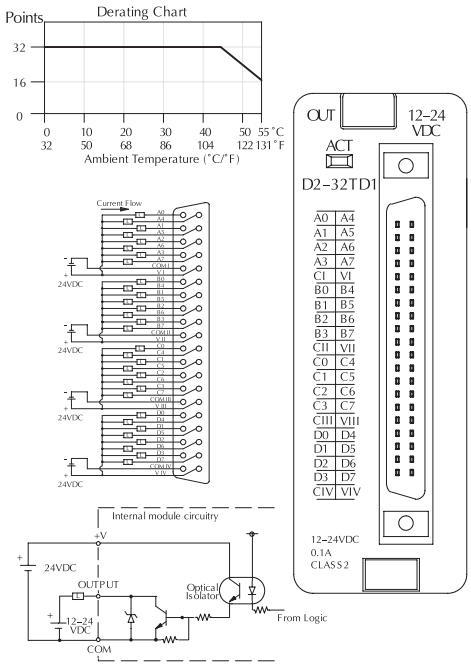


When the A/B switch is in the A position, the LEDs display the output status of the module's first 8 output points. Positon B displays the output status of the module's second group of 8 output points.

D2–32TD1, DC Output

D2-32TD1 DC Output	
Outputs per Module	32 (current sinking)
Commons per Module	4 (8 I/O terminal points)
Output Type	NPN open collector
Operating Voltage	12–24 VDC
Peak Voltage	30VDC
ON Voltage Drop	0.5 VDC maximum
Minimum Load Current	0.2 mA
Max Load Current	0.1 A/point; 3.2 A per module
Max Leakage Current	0.1 mA @ 30VDC
Max Inrush Current	150mA for 10ms
Base Power Required 5VDC	350mA
OFF to ON Response	0.5 ms
ON to OFF Response	0.5 ms
Terminal Type (not included)	40-pin connector[1]
Status Indicator	Module activity (no I/O status indicators)
Weight	2.1 oz. (60g)
Fuses	None
External DC Power Required	20–28 VDC max. 120mA (all points on)

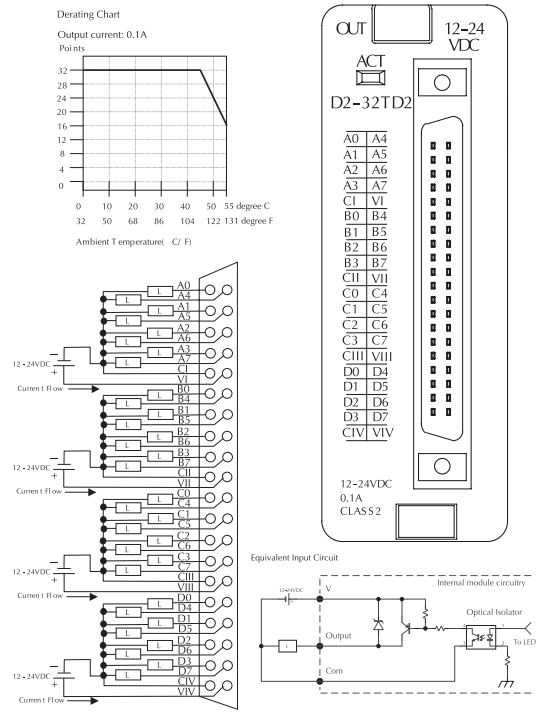
[1] Connector sold separately.
See Terminal Blocks and Wiring for wiring options.



D2–32TD2, DC Output

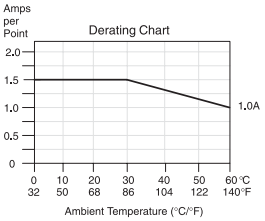
D2-32TD2 DC Output	
Outputs per Module	32 (current sourcing)
Commons per Module	4 (8 I/O terminal points)
Output Type	Transistor
Operating Voltage	12 to 24 VDC
Peak Voltage	30VDC
ON Voltage Drop	0.5 VDC @ 0.1 A
Minimum Load Current	0.2 mA
Max Load Current	0.1 A/point; 0.8 A/common
Max Leakage Current	0.1 mA @ 30VDC
Max Inrush Current	150mA @ 10ms
Base Power Required 5VDC	350mA
OFF to ON Response	0.5 ms
ON to OFF Response	0.5 ms
Terminal Type (not included)	40-pin connector[1]
Status Indicator	Module activity (no I/O status indicators)
Weight	2.1 oz (60g)
Fuses	None

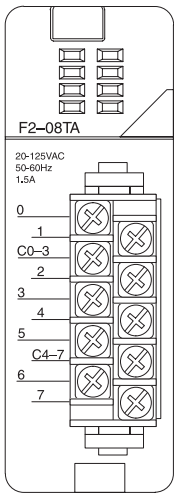
[1] Connector sold separately.
See Terminal Blocks and Wiring for wiring options.

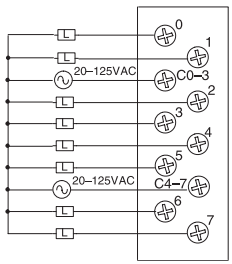


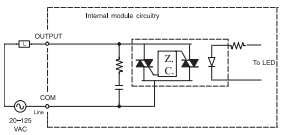
F2–08TA, AC Output

F2-08TA AC Output	
Outputs per Module	8
Commons per Module	2 (Isolated)
Output Type	SSR (Triac with zero crossover)
Operating Voltage	24–140 VAC
Peak Voltage	140VAC
ON Voltage Drop	1.6 V(rms) @ 1.5 A
AC Frequency	47 to 63 Hz
Minimum Load Current	50mA
Max Load Current	1.5 A / pt @ 30ºC 1.0 A / pt @ 60ºC 4.0 A / common; 8.0 A / module @ 60ºC
Max Leakage Current	0.7 mA (rms)
Peak One Cycle Surge Current	15A
Base Power Required 5VDC	250mA
OFF to ON Response	0.5 ms - 1/2 cycle
ON to OFF Response	0.5 ms - 1/2 cycle
Terminal Type (included)	Removable; D2-8IOCON
Status Indicator	Logic side
Weight	3.5 oz. (100g)
Fuses	None



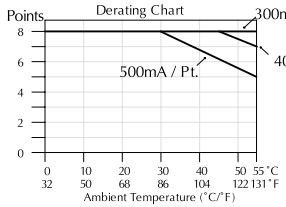


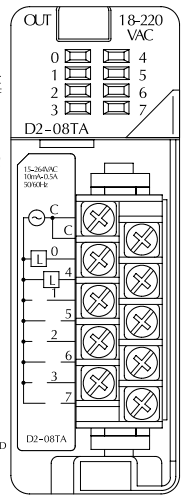


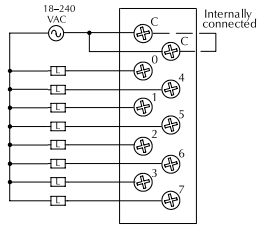


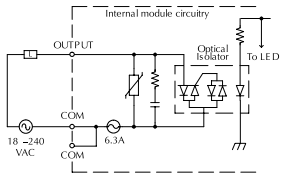
D2–08TA, AC Output

D2-08TA AC Output	
Outputs per Module	8
Commons per Module	1 (2 I/O terminal points)
Output Type	SSR (Triac)
Operating Voltage	15–264 VAC
Peak Voltage	264VAC
ON Voltage Drop	< 1.5 VAC (>0.1 A) < 3.0 VAC (<0.1 A)
AC Frequency	47 to 63Hz
Minimum Load Current	10mA
Max Load Current	0.5 A/point; 4A/common
Max Leakage Current	4mA (264VAC, 60Hz) 1.2 mA (100VAC, 60Hz) 0.9 mA (100VAC, 50Hz)
Max Inrush Current	10A for 10ms
Base Power Required 5VDC	250mA
OFF to ON Response	1 ms
ON to OFF Response	1 ms + 1/2 cycle
Terminal Type (included)	Removable; D2-8IOCON
Status Indicator	Logic side
Weight	2.8 oz. (80g)
Fuses	1 per common, 6.3 A slow blow, non-replaceable





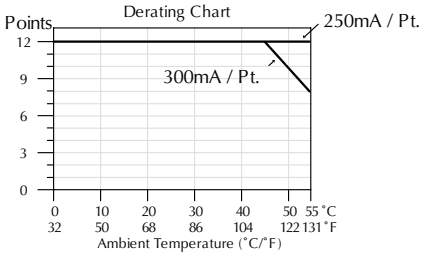


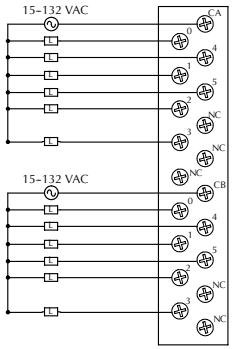


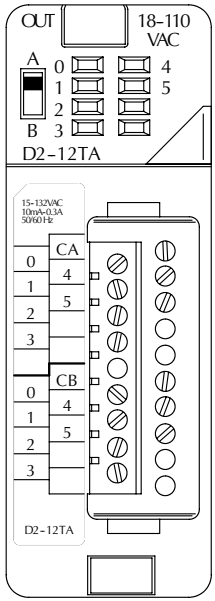
D2–12TA, AC Output

D2-12TA AC Output	
Outputs per Module	12
Outputs Points Consumed	16 (four unused, see chart below)
Commons per Module	2 (isolated)
Output Type	SSR (Triac)
Operating Voltage	15–132 VAC
Peak Voltage	132VAC
ON Voltage Drop	< 1.5 VAC (>50mA) < 4.0 VAC (<50mA)
AC Frequency	47 to 63 Hz
Minimum Load Current	10mA
Max Load Current	0.3 A/point; 1.8 A/common

Max Leakage Current	2mA (132VAC, 60Hz)
Max Inrush Current	10A for 10ms
Base Power Required 5VDC	350mA
OFF to ON Response	1ms
ON to OFF Response	1ms + 1/2 cycle
Terminal Type (included)	Removable; D2-16IOCON
Status Indicator	Logic side
Weight	2.8 oz. (80g)
Fuses	(2) 1 per common 3.15 A slow blow, replaceable Order D2-FUSE-1 (5 per pack)



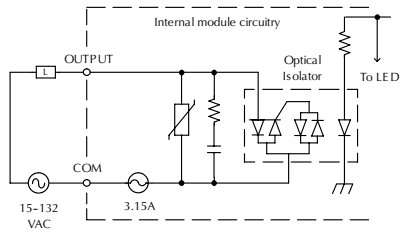




Addresses Used

Points	Used?	Points	Used?
Yn+0	Yes	Yn+10	Yes
Yn+1	Yes	Yn+11	Yes
Yn+2	Yes	Yn+12	Yes
Yn+3	Yes	Yn+13	Yes
Yn+4	Yes	Yn+14	Yes
Yn+5	Yes	Yn+15	Yes
Yn+6	No	Yn+16	No
Yn+7	No	Yn+17	No

n is the starting address

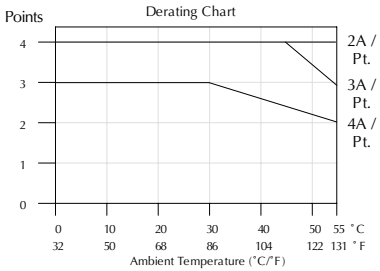


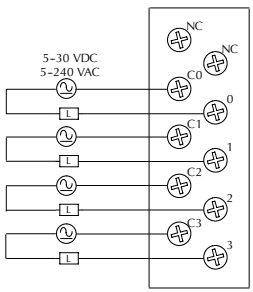
D2–04TRS, Relay Output

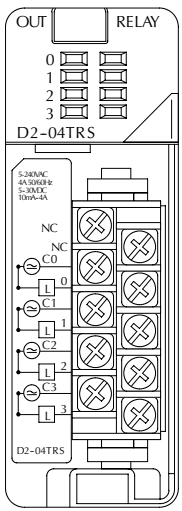
D2-04TRS Relay Output	
Outputs per Module	4
Outputs Points Consumed	8 (only 1st 4pts. are used)
Commons per Module	4 (isolated)
Output Type	Relay, form A (SPST)
Operating Voltage	5-30 VDC / 5-240 VAC
Peak Voltage	30 VDC, 264 VAC
ON Voltage Drop	0.72 VDC maximum
AC Frequency	47 to 63 Hz
Minimum Load Current	10mA
Max Load Current (resistive)	4A/point; 8A/module (resistive)

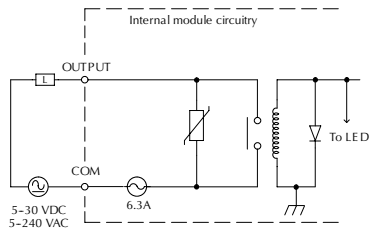
Max Leakage Current	0.1 mA @ 264VAC
Max Inrush Current	5A for < 10ms
Base Power Required 5VDC	250mA
OFF to ON Response	10ms
ON to OFF Response	10ms
Terminal Type (included)	Removable; D2-8IOCON
Status Indicator	Logic side
Weight	2.8 oz. (80g)
Fuses	1 per point 6.3 A slow blow, replaceable Order D2-FUSE-3 (5 per pack)

Typical Relay Life (Operations)				
	Voltage & Load Current			
Type of Load	1A	2A	3A	4A
24VDC Resistive	500K	200K	100K	50K
24VDC Solenoid	100K	40K	–	–
110VAC Resistive	500K	250K	150K	100K
110VAC Solenoid	200K	100K	50K	–
220VAC Resistive	350K	150K	100K	50K
220VAC Solenoid	100K	50K	–	–
At 24VDC, solenoid (inductive) loads over 2A cannot be used.				
At 100VAC, solenoid (inductive) loads over 3A cannot be used.				
At 220VAC, solenoid (inductive) loads over 2A cannot be used.				







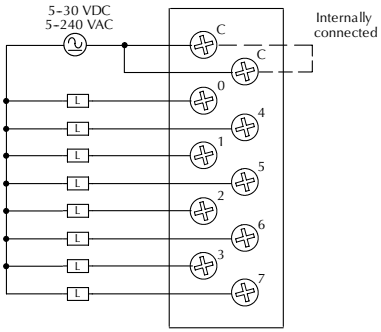


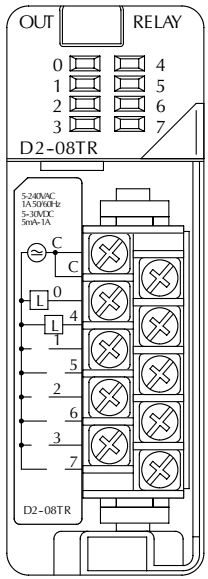
D2–08TR, Relay Output

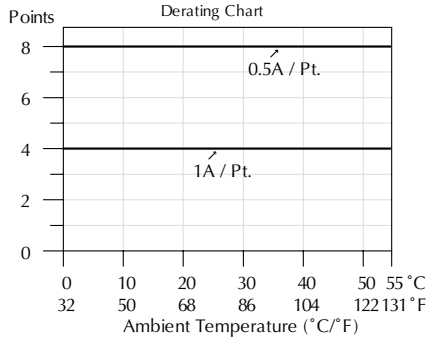
D2-08TR Relay Output	
Outputs per Module	8
Outputs Points Consumed	8
Commons per Module	1 (2 I/O terminals)
Output Type	Relay, form A (SPST)
Operating Voltage	5–30 VDC; 5–240 VAC
Peak Voltage	30VDC, 264VAC
ON Voltage Drop	N/A
AC Frequency	47 to 60 Hz
Minimum Load Current	5mA @ 5VDC
Max Load Current (resistive)	1A/point; 4A/common

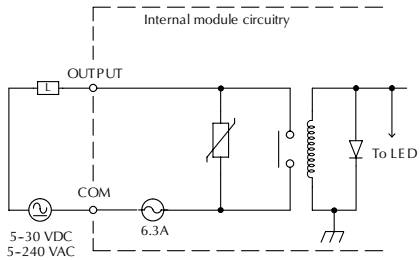
Max Leakage Current	0.1 mA @265 VAC
Max Inrush Current	Output: 3A for 10ms Common: 10A for 10ms
Base Power Required 5VDC	250mA
OFF to ON Response	12ms
ON to OFF Response	10ms
Terminal Type (included)	Removable; D2-8IOCON
Status Indicator	Logic side
Weight	3.9 oz. (110g)
Fuses	One 6.3A slow blow, replaceable Order D2-FUSE-3 (5 per pack)

Typical Relay Life (Operations)		
Voltage/Load	Current	Closures
24VDC Resistive	1A	500K
24VDC Solenoid	1A	100K
110VAC Resistive	1A	500K
110VAC Solenoid	1A	200K
220VAC Resistive	1A	350K
220VAC Solenoid	1A	100K









F2–08TR, Relay Output

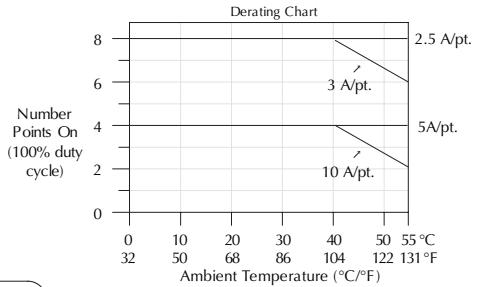
F2-08TR Relay Output	
Outputs per Module	8
Outputs Points Consumed	8
Commons per Module	2 (isolated), 4-pts. per common
Output Type	8, Form A (SPST normally open)
Operating Voltage	7A @ 12–28 VDC, 12–250 VAC; 0.5 A @ 120VDC
Peak Voltage	150VDC, 265VAC
ON Voltage Drop	N/A
AC Frequency	47 to 63 Hz
Minimum Load Current	10 mA @ 12 VDC
Max Load Current (resistive)	10A/point [3] (subject to derating) Max of 10A/common
Max Leakage Current	N/A
Max Inrush Current	12A
Base Power Required 5VDC	670mA
OFF to ON Response	15ms (typical)
ON to OFF Response	5ms (typical)
Terminal Type (included)	Removable; D2-8IOCON
Status Indicator	Logic side
Weight	5.5 oz. (156g)
Fuses	None

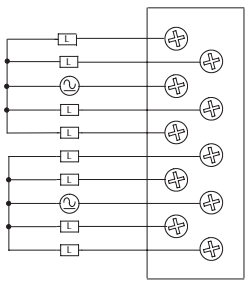
Typical Relay Life (Operations) at Room Temperature			
	Voltage & Load Current		
Type of Load	50mA	5A	7A
24VDC Resistive	10M	600K	300K
24VDC Solenoid	–	150K	75K
110VAC Resistive	–	600K	300K
110VAC Solenoid	–	500K	200K
220VAC Resistive	–	300K	150K
220VAC Solenoid	–	250K	100K

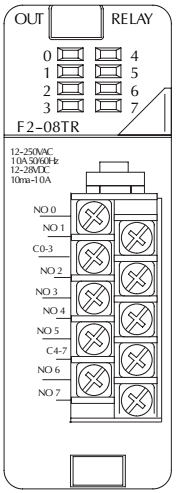
1) Contact life may be extended beyond those values shown with the use of arc suppression techniques described in the DL205 User Manual. Since these modules have no leakage current, they do not have built-in snubber. For example, if you place a diode across a 24VDC inductive load, you can significantly increase the life of the relay.

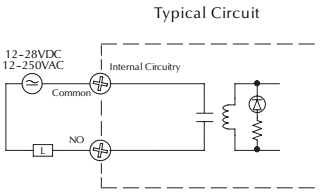
2) At 120VDC 0.5 A resistive load, contact life cycle is 200K cycles.

3) Normally closed contacts have 1/2 the current handling capability of the normally open contacts.









F2–08TRS, Relay Output

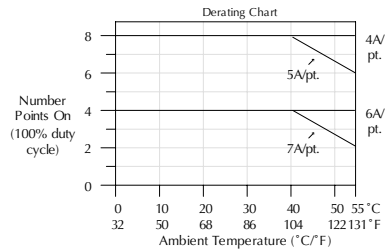
F2-08TRS Relay Output	
Outputs per Module	8
Outputs Points Consumed	8
Commons per Module	8 (isolated)
Output Type	3, Form C (SPDT) 5, Form A (SPST normally open)
Operating Voltage	7A @ 12–28 VDC, 12–250 VAC 0.5A @ 120VDC
Peak Voltage	150VDC, 265VAC
ON Voltage Drop	N/A
AC Frequency	47 to 63Hz
Minimum Load Current	10mA @ 12VDC
Max Load Current (resistive)	7A/point [3] (subject to derating)
Max Leakage Current	N/A
Max Inrush Current	12A
Base Power Required 5VDC	670mA
OFF to ON Response	15ms (typical)
ON to OFF Response	5ms (typical)
Terminal Type (included)	Removable; D2-16IOCON
Status Indicator	Logic side
Weight	5.5 oz. (156g)
Fuses	None

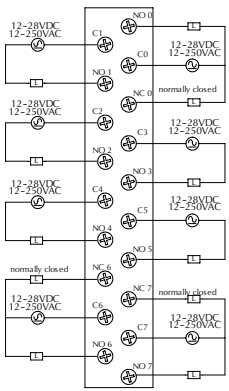
Typical Relay Life (Operations) at Room Temperature			
	Voltage & Load Current		
Type of Load	50mA	5A	7A
24VDC Resistive	10M	600K	300K
24VDC Solenoid	–	150K	75K
110VAC Resistive	–	600K	300K
110VAC Solenoid	–	500K	200K
220VAC Resistive	–	300K	150K
220VAC Solenoid	–	250K	100K

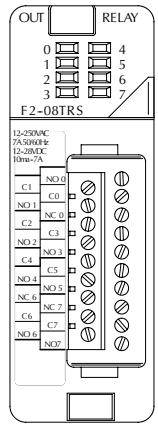
1) Contact life may be extended beyond those values shown with the use of arc suppression techniques described in the DL205 User Manual. Since these modules have no leakage current, they do not have built-in snubber. For example, if you place a diode across a 24VDC inductive load, you can significantly increase the life of the relay.

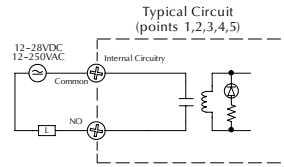
2) At 120VDC 0.5 A resistive load, contact life cycle is 200K cycles.

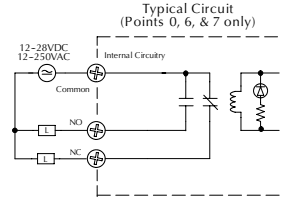
3) Normally closed contacts have 1/2 the current handling capability of the normally open contacts.









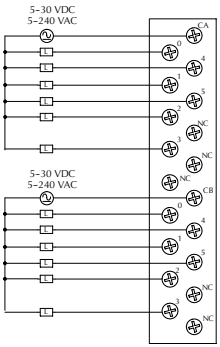


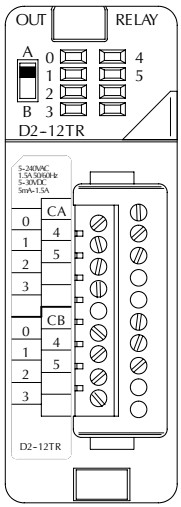
D2–12TR, Relay Output

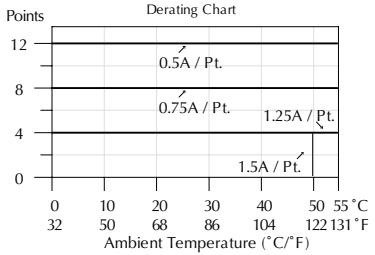
D2-12TR Relay Output	
Outputs per Module	12
Outputs Points Consumed	16 (four unused, see chart below)
Commons per Module	2 (6-pts. per common)
Output Type	Relay, form A (SPST)
Operating Voltage	5–30 VDC; 5–240 VAC
Peak Voltage	30VDC; 264VAC
ON Voltage Drop	N/A
AC Frequency	47 to 60 Hz
Minimum Load Current	5mA @ 5VDC
Max Load Current (resistive)	1.5 A/point; Max of 3A/common
Max Leakage Current	0.1 mA @ 265VAC
Max Inrush Current	Output: 3A for 10ms Common: 10A for 10ms
Base Power Required 5VDC	450mA
OFF to ON Response	10ms
ON to OFF Response	10ms
Terminal Type (included)	Removable; D2-16IOCON
Status Indicator	Logic side
Weight	4.6 oz. (130g)
Fuses	(2) 4A slow blow, replaceable Order D2-FUSE-4 (5 per pack)

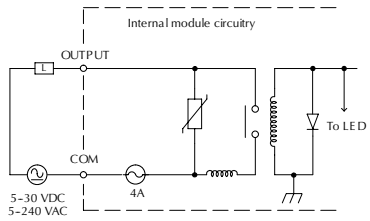
Typical Relay Life (Operations)		
Voltage/Load	Current	Closures
24VDC Resistive	1A	500K
24VDC Solenoid	1A	100K
110VAC Resistive	1A	500K
110VAC Solenoid	1A	200K
220VAC Resistive	1A	350K
220VAC Solenoid	1A	100K

Addresses Used			
Points	*Used?*	*Points*	*Used?*
Yn+0	Yes	Yn+10	Yes
Yn+1	Yes	Yn+11	Yes
Yn+2	Yes	Yn+12	Yes
Yn+3	Yes	Yn+13	Yes
Yn+4	Yes	Yn+14	Yes
Yn+5	Yes	Yn+15	Yes
Yn+6	No	Yn+16	No
Yn+7	No	Yn+17	No
n is the starting address			







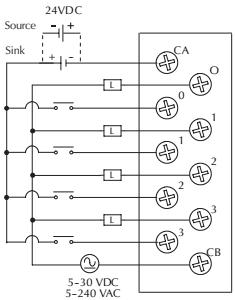


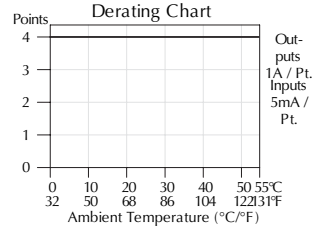
D2–08CDR, 4 pt. DC Input / 4 pt. Relay Output

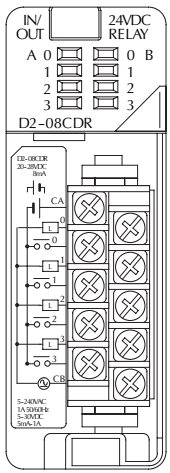
D2-08CDR 4-pt. DC In / 4pt. Relay Out	
General Specifications	
Base Power Required 5VDC	200mA
Terminal Type (included)	Removable; D2-8IOCON
Status Indicator	Logic side
Weight	3.5 oz. (100 g)
Input Specifications	
Inputs per Module	4 (sink/source)
Input Points Consumed	8 (only first 4-pts. are used)
Commons per Module	1
Input Voltage Range	20–28 VDC
Peak Voltage	30VDC
ON Voltage Level	19VDC minimum
OFF Voltage Level	7VDC maximum
AC Frequency	N/A
Input Impedance	4.7 kΩ
Input Current	5mA @ 24VDC
Maximum Current	8mA @ 30VDC
Minimum ON Current	4.5 mA
Maximum OFF Current	1.5 mA
OFF to ON Response	1 to 10 ms
ON to OFF Response	1 to 10 ms
Fuses (input circuits)	None

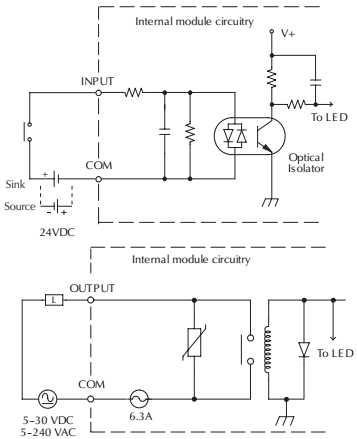
Output Specifications	
Outputs per Module	4
Outputs Points Consumed	8 (only first 4-pts. are used)
Commons per Module	1
Output Type	Relay, form A (SPST)
Operating Voltage	5–30 VDC; 5–240 VAC
Peak Voltage	30VDC; 264VAC
ON Voltage Drop	N/A
AC Frequency	47 to 63 Hz
Minimum Load Current	5mA @ 5VDC
Max Load Current (resistive)	1A/point ; 4A/module
Max Leakage Current	0.1 mA @ 264VAC
Max Inrush Current	3A for < 100ms 10A for < 10ms (common)
OFF to ON Response	12ms
ON to OFF Response	10ms
Fuses (output circuits)	1 (6.3 A slow blow, replaceable); Order D2-FUSE-3 (5 per pack)

Typical Relay Life (Operations)		
Voltage/Load	**Current**	**Closures**
24VDC Resistive	1A	500K
24VDC Solenoid	1A	100K
110VAC Resistive	1A	500K
110VAC Solenoid	1A	200K
220VAC Resistive	1A	350K
220VAC Solenoid	1A	100K









Glossary of Specification Terms

Inputs or Outputs Per Module
Indicates number of input or output points per module and designates current sinking, current sourcing, or either.

Commons Per Module
Number of commons per module and their electrical characteristics.

Input Voltage Range
The operating voltage range of the input circuit.

Output Voltage Range
The operating voltage range of the output circuit.

Peak Voltage
Maximum voltage allowed for the input circuit.

AC Frequency
AC modules are designed to operate within a specific frequency range.

ON Voltage Level
The voltage level at which the input point will turn ON.

OFF Voltage Level
The voltage level at which the input point will turn OFF.

Input impedance
Input impedance can be used to calculate input current for a particular operating voltage.

Input Current
Typical operating current for an active (ON) input.

Minimum ON Current
The minimum current for the input circuit to operate reliably in the ON state.

Maximum OFF Current
The maximum current for the input circuit to operate reliably in the OFF state.

Minimum Load
The minimum load current for the output circuit to operate properly.

External DC Required
Some output modules require external power for the output circuitry.

ON Voltage Drop
Sometimes called “saturation voltage,” it is the voltage measured from an output point to its common terminal when the output is ON at maximum load.

Maximum Leakage Current

The maximum current a connected maximum load will receive when the output point is OFF.

Maximum Inrush Current

The maximum current used by a load for a short duration upon an OFF to ON transition of an output point. It is greater than the normal ON state current and is characteristic of inductive loads in AC circuits.

Base Power Required

Power from the base power supply is used by the DL205 input modules and varies between different modules. The guidelines for using module power are explained in the power budget configuration section in Chapter 4–7.

OFF to ON Response

The time the module requires to process an OFF to ON state transition.

ON to OFF Response

The time the module requires to process an ON to OFF state transition.

Terminal Type

Indicates whether the terminal type is a removable or non-removable connector or a terminal.

Status Indicators

The LEDs that indicate the ON/OFF status of an input point. These LEDs are electrically located on either the logic side or the field device side of the input circuit.

Weight

Indicates the weight of the module. See Appendix F for a list of the weights for the various DL205 components.

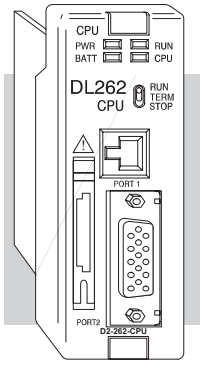
Fuses

Protective devices for an output circuit, which stop current flow when current exceeds the fuse rating. They may be replaceable or non–replaceable, or located externally or internally.

CPU Specifications and Operations

In This Chapter...

CPU Overview



The Central Processing Unit is the heart of the PLC. Almost all system operations are controlled by the CPU, so it is important that it is set up and installed correctly. This chapter provides the information needed to understand:

- The differences between the various models of CPUs, and
- The steps required to set up and install the CPU.

General CPU Features

The D2-230, D2-240, D2-250–1, D2–260 and D2-262 are modular CPUs which can be installed in 3, 4, 6, or 9 slot bases.

A large selection of I/O modules in the DL205 family, provides flexibility when building a system (See Chapter 4). The DL205 CPUs offer a wide range of processing power and program instructions. All offer RLL and Stage program instructions (See Chapter 5). They also provide extensive internal diagnostics that can be monitored from the application program or from an operator interface.

D2-230 CPU Features

The D2-230 has 2.4K words of memory comprised of 2.0K of ladder memory and approximately 400 words of V-memory (data registers). It has 92 different instructions available for programming, and supports a maximum of 256 I/O points.

Program storage is in the factory-installed EEPROM. In addition to the EEPROM there is also RAM on the CPU which will store system parameters, V-memory, and other data which is not in the application program.

The D2-230 provides one built-in RS-232 communication port, so you can easily connect a handheld programmer or a personal computer without needing any additional hardware.

D2-240 CPU Features

The D2-240 has a maximum of 3.8K of memory comprised of 2.5K of ladder memory and approximately 1.3K of V-memory (data registers). There are 129 instructions available for program development and a maximum of 256 points local I/O, and 896 points with remote I/O are supported.

Program storage is in the factory-installed EEPROM. In addition to the EEPROM, there is also RAM on the CPU that will store system parameters, V-memory and other data which is not in the application program.

The D2-240 has two communication ports. The top port is the same port configuration as the D2-230. The bottom port also supports the ***Direct***NET protocol, so you can use the D2-240 in a ***Direct***NET network. Since the port is RS-232, you must use an RS-232/RS-422 converter for multi-drop connections.

D2-250–1 CPU Features

The D2-250–1 replaces the D2-250 CPU. It offers all the D2-240 features, plus more program instructions and a built–in Remote I/O Master port. It offers all the features of the D2-250 CPU with the addition of supporting Local expansion I/O. It has a maximum of 14.8K of program memory comprised of 7.6K of ladder memory and 7.2K of V-memory (data registers). It supports a maximum of 256 points of local I/O and a maximum of 768 I/O points (maximum of two local expansion bases). In addition, port 2 supports up to 2048 points if you use the D2-250–1 as a Remote master. It includes an internal RISC–based microprocessor for greater processing power. The D2-250–1 has 240 instructions. The instructions are in addition to the D2-240 instruction set which includes drum timers, a print function, floating point math, PID loop control for 4 loops and the Intelligent Box (IBox) instructions.

The D2-250–1 has a total of two built–in communications ports. The top port is identical to the top port of the D2-240, with the exception of the ***Direct***Net slave feature. The bottom port is a 15–pin RS-232/RS-422 port. It will interface with ***Direct***SOFT and operator interfaces, and provides ***Direct***Net and Modbus RTU Master/Slave connections.

D2-260 and D2-262 CPU Features

The D2-260 and D2-262 offer all the D2-250–1 features, plus ASCII IN/OUT and expanded Modbus instructions. They support up to 1280 local I/O points by using up to four local expansion bases. They have a maximum of 30.4K of program memory comprised of 15.8K of ladder memory (saved on flash memory) and 14.6K of V-memory (data registers). They also include an internal RISC–based microprocessor for greater processing power. The D2-260 and D2-262 have 297 instructions. In addition to the D2-250–1 instruction set, the D2-260 and D2-262 instruction set includes table instructions, trigonometric instructions and support for 16 PID loops.

The D2-260 and D2-262 each have two built–in communications ports. The top port is identical to the top port of the D2-250–1. The bottom port is a 15–pin RS-232/RS-422/RS-485 port. It will interface with ***Direct***SOFT, operator interfaces, and provides ***Direct***Net, Modbus RTU Master/Slave connections. Port 2 also supports ASCII IN/OUT instructions.

CPU General Specifications

Feature	D2-230	D2-240	D2-250–1	D2-260/ D2-262
Total Program memory (words)	2.4K	3.8K	14.8K	30.4K
Ladder memory (words)	2048	2560	7680 (Flash)	15872 (Flash)
V-memory (words)	256	1024	7168	14592
Non-volatile V Memory (words)	128	256	No	No
Boolean execution /K	4–6 ms	10–12 ms	1.9 ms	1.9 ms/1ms
RLL and RLL^{PLUS} Programming	Yes	Yes	Yes	Yes
Handheld programmer	Yes	Yes	Yes	Yes
*Direct*SOFT programming for Windows.	Yes	Yes	Yes	Yes
Built-in communication ports	One RS–232	Two RS–232	One RS–232 One RS–232 or RS–422	One RS–232 One RS–232, RS–422 or RS–485
EEPROM	Standard on CPU	Standard on CPU	Flash	Flash
Total CPU memory I/O points available	256 (X,Y,CR)	896 (X, Y, CR)	2048 (X, Y, CR)	8192 (X, Y, CR, GX, GY)
Local I/O points available	256	256	256	256
Local Expansion I/O points (including local I/O and expansion I/O points)	N/A	N/A	768 (2 exp. bases max.)	1280 (4 exp. bases max.)
Serial Remote I/O points (including local I/O and expansion I/O points)	N/A	896	2048	8192
Serial Remote I/O Channels	N/A	2	8	8
Max Number of Serial Remote Slaves	N/A	7 Remote / 31 Slice	7 Remote / 31 Slice	7 Remote / 31 Slice
Ethernet Remote I/O Discrete points	N/A	896	2048	8192
Ethernet Remote I/O Analog I/O channels	N/A	Map into V–memory	Map into V–memory	Map into V–memory
Ethernet Remote I/O channels	N/A	Limited by power budget	Limited by power budget	Limited by power budget
Max Number of Ethernet slaves per channel	N/A	16	16	16
I/O points per Remote channel	N/A	16,384 (limited to 896 by CPU)	16,384 (16 fully expanded H4–EBC slaves using V–memory and bit–of–word instructions)	16,384 (16 fully expanded H4–EBC slaves using V–memory and bit–of–word instructions
I/O Module Point Density	4/8/12/16/32	4/8/12/16/32	4/8/12/16/32	4/8/12/16/32
Slots per Base	3/4/6/9	3/4/6/9	3/4/6/9	3/4/6/9

Feature	D2-230	D2-240	D2-250–1	D2-260/D2-262
Number of instructions available (see Chapter 5 for details)	92	129	240	297
Control relays	256	256	1024	2048
Special relays (system defined)	112	144	144	144
Stages in RLL^{PLUS}	256	512	1024	1024
Timers	64	128	256	256
Counters	64	128	128	256
Immediate I/O	Yes	Yes	Yes	Yes
Interrupt input (hardware / timed)	Yes / No	Yes / Yes	Yes / Yes	Yes / Yes
Subroutines	No	Yes	Yes	Yes
Drum Timers	No	No	Yes	Yes
Table Instructions	No	No	No	Yes
For/Next Loops	No	Yes	Yes	Yes
Math	Integer	Integer	Integer, Floating Point	Integer, Floating Point, Trigonometric
ASCII	No	No	Yes, OUT	Yes, IN/OUT
PID Loop Control, Built In	No	No	Yes, 4 Loops	Yes, 16 Loops
Time of Day Clock/Calendar	No	Yes	Yes	Yes
Run Time Edits	Yes	Yes	Yes	Yes
Supports Overrides	No	Yes	Yes	Yes
Internal diagnostics	Yes	Yes	Yes	Yes
Password security	Yes	Yes	Yes	Yes
System error log	No	Yes	Yes	Yes
User error log	No	Yes	Yes	Yes
Battery backup	Yes (optional)	Yes (optional)	Yes (optional)	Yes (optional)

CPU Base Electrical Specifications

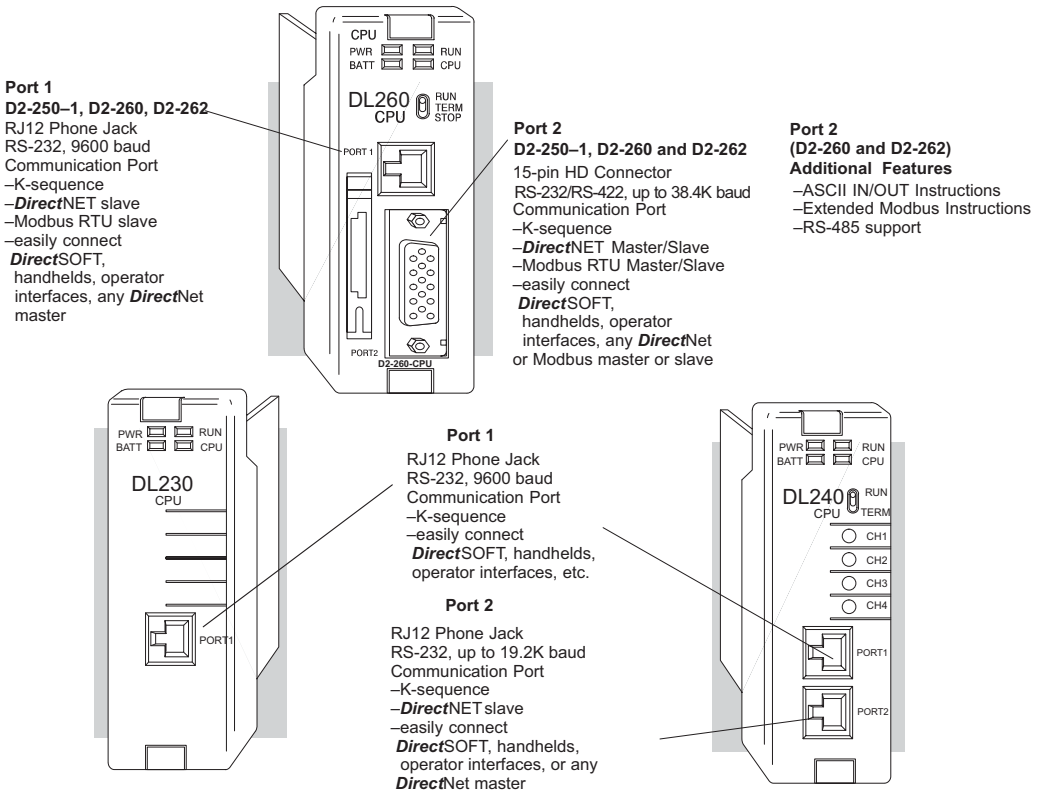
Specification	AC Powered Bases	24 VDC Powered Bases	125 VDC Powered Bases
Part Numbers	D2–03B–1 D2–04B–1 D2–06B–1 D2–09B–1	D2–03BDC1–1 D2–04BDC1–1 D2–06BDC1–1 D2–09BDC1–1	D2–06BDC2–1 D2–09BDC2–1
Input Voltage Range	100–240 VAC +10% –15%	10.2–28.8 VDC (24VDC) with less than 10% ripple	104–240 VDC +10% –15%
Maximum Inrush Current	30A	10A	20A
Maximum Power	80VA	25W	30W
Voltage Withstand (dielectric)	1 minute @ 1500VAC between primary, secondary, field ground, and run relay		
Insulation Resistance	> 10MΩ at 500VDC		
Auxiliary 24 VDC Output	20–28 VDC, less than 1V p-p 300mA max.	None	20–28 VDC, less than 1V p-p 300mA max.
Fusing (internal to base power supply)	Non–replaceable 2A @ 250V slow blow fuse	Non–replaceable 3.15 A @ 250V slow blow fuse	Non–replaceable 2A @ 250V slow blow fuse

CPU Hardware Setup

Communication Port Pinout Diagrams

Cables are available that allow you to quickly and easily connect a Handheld Programmer or a personal computer to the DL205 CPUs. However, if you need to build a cable(s), use the pinout descriptions shown on the following pages. You can also use the Tech Support/Cable Wiring diagrams located on our website.

The D2-240, D2-250–1, D2-260 and D2-262 CPUs have two ports while the D2-230 has only one. All of the CPUs require at least one RJ-12 connector. The D2-250-1, D2-260 and D2-262 require one 15 pin D-shell connector.

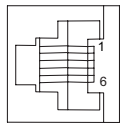


Port 1 Specifications (D2-230 and D2-240 CPUs)

☑ 230
☑ 240
☒ 250-1
☒ 260
☒ 262

The operating parameters for Port 1 on the D2-230 and D2-240 CPUs are fixed.

- 6-pin female modular (RJ12 phone jack) type connector
- K–sequence protocol (slave only)
- RS-232, 9600 baud
- Connect to *Direct*SOFT, D2–HPP, DV–1000, HMI panels
- Fixed station address of 1
- 8 data bits, one stop
- Asynchronous, Half–duplex, DTE
- Odd parity



6-pin Female
Modular Connector

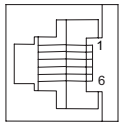
Port 1 Pin Descriptions (D2-230 and D2-240)		
1	0V	Power (–) connection (GND)
2	5V	Power (+) connection
3	RXD	Receive Data (RS-232)
4	TXD	Transmit Data (RS-232)
5	5V	Power (+) connection
6	0V	Power (–) connection (GND)

Port 1 Specifications (D2-250–1, D2-260 and D2-262 CPUs)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The operating parameters for Port 1 on the D2-250–1, D2-260 and D2-262 CPUs are fixed. This applies to the D2-250 as well.

- 6-pin female modular (RJ12 phone jack) type connector
- K–sequence protocol (slave only)
- *Direct*NET (slave only)
- Modbus RTU (slave only) - supported only on D2-250-1, D2-260 and D2-262 CPUs
- RS-232, 9600 baud
- Connect to *Direct*SOFT, D2–HPP, DV1000 or *Direct*NET master
- 8 data bits, one start, one stop
- Asynchronous, Half–duplex, DTE
- Odd parity



6-pin Female
Modular Connector

Port 1 Pin Descriptions (D2-250-1, D2-260 and D2-262)		
1	0V	Power (–) connection (GND)
2	5V	Power (+) connection
3	RXD	Receive Data (RS-232C)
4	TXD	Transmit Data (RS-232C
5	5V	Power (+) connection
6	0V	Power (–) connection (GND)

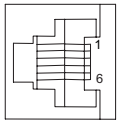
NOTE: *The 5V pins are rated at 200mA maximum, primarily for use with some operator interface units.*

Port 2 Specifications (D2-240)

☒ 230
☑ 240
☒ 250-1
☒ 260
☒ 262

The operating parameters for Port 2 on the D2-240 CPU are configurable using Aux functions on a programming device.

- 6-Pin female modular (RJ12 phone jack) type connector
- K–sequence protocol, *Direct*NET (slave),
- RS-232, Up to 19.2K baud
- Address selectable (1–90)
- Connect to *Direct*SOFT, D2–HPP, DV-1000, HMI, or *Direct*NET master
- 8 data bits, one start, one stop
- Asynchronous, Half–duplex, DTE
- Odd or no parity



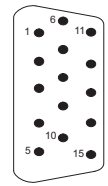
6-pin Female
Modular Connector

Port 2 Pin Descriptions (D2-240)		
1	0V	Power (–) connection (GND)
2	5V	Power (+) connection
3	RXD	Receive Data (RS-232)
4	TXD	Transmit Data (RS-232)
5	RTS	Request to Send
6	0V	Power (–) connection (GND)

Port 2 Specifications (D2-250–1, D2-260 and D2-262)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

Port 2 on the D2-250-1, D2-260 and D2-262 CPUs is located on the 15-pin D-shell connector. It is configurable using AUX functions on a programming device. This applies to the D2-250 as well.



15-pin Female D Connector

- 15-Pin female D type connector
- Protocol: K-sequence (Slave only), *Direct*NET Master/Slave, Modbus RTU Master/Slave, Remote I/O, (ASCII IN/OUT D2-260 and D2-262 only)
- RS-232, non-isolated, distance within 15m (approximately 50ft)
- RS-422, non-isolated, distance within 1000m (approximately 3280ft)
- RS-485, non–isolated, distance within 1000m (D2-260 and D2-262 only)
- Up to 38.4 Kbaud (D2-250(-1), D2-260); 2400 to 38.4Kbaud (D2-262)
- Address selectable (1–90)
- Connects to *Direct*SOFT, D2–HPP, operator interfaces, any *Direct*NET or Modbus Master/Slave, (ASCII devices-D2-260 and D2-262 only)
- 8 data bits, one start, one stop
- Asynchronous, Half–duplex, DTE Remote I/O
- Odd/even/none parity

Port 2 Pin Descriptions (D2-250–1, D2-260 and D2-262)		
1	5V	5VDC
2	TXD2	Transmit Data (RS-232)
3	RXD2	Receive Data (RS-232)
4	RTS2	Ready to Send (RS–232)
5	CTS2	Clear to Send (RS–232)
6	RXD2 –	Receive Data – (RS–422) (RS–485 D2-260/ D2-262)
7	0V	Logic Ground
8	0V	Logic Ground
9	TXD2 +	Transmit Data + (RS–422) (RS–485 D2-260/ D2-262)
10	TXD2 –	Transmit Data – (RS–422) (RS–485 D2-260/ D2-262)
11	RTS2 +	Request to Send + (RS–422) (RS–485 D2-260/ D2-262)
12	RTS2 –	Request to Send – (RS–422)(RS–485 D2-260/ D2-262)
13	RXD2 +	Receive Data + (RS–422) (RS–485 D2-260/ D2-262)
14	CTS2 +	Clear to Send + (RS422) (RS–485 D2-260/ D2-262)
15	CTS2 –	Clear to Send – (RS–422) (RS–485 D2-260/ D2-262)

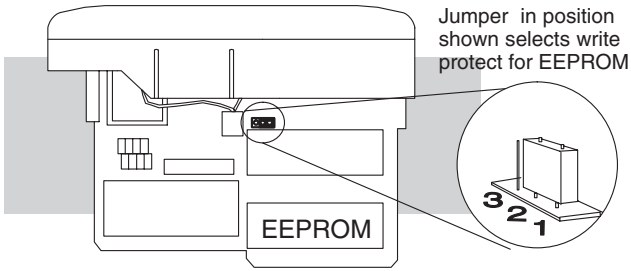
Selecting the Program Storage Media

Built-in EEPROM

☑ 230
☑ 240
☒ 250-1
☒ 260
☒ 262

The D2-230 and D2-240 CPUs provide built-in EEPROM storage. This type of memory is non-volatile and is not dependent on battery backup to retain the program. The EEPROM can be electrically reprogrammed without being removed from the CPU. You can also set Jumper 3, which will write protect the EEPROM. The jumper is set at the factory to allow changes to EEPROM. If you select write protection by changing the jumper position, you cannot make changes to the program.

WARNING: Do NOT change Jumper 2. This is for factory test operations. If you change Jumper 2, the CPU will not operate properly.



EEPROM Sizes

The D2-230 and D2-240 CPUs use different sizes of EEPROMs. The CPUs come from the factory with EEPROMs already installed. However, if you need extra EEPROMs, select one that is compatible with the following part numbers.

CPU Type	EEPROM Part Number	Capacity
D2-230	Hitachi HN58C65P–25	8K byte (2Kw)
D2-240	Hitachi HN58C256P–20	32K byte (3Kw)

EEPROM Operations

Many AUX functions are specifically for use with an EEPROM in the Handheld Programmer. This enables you to quickly and easily copy programs between a program developed offline in the Handheld Programmer and the CPU. Also, you can erase EEPROMs, compare them, etc. See the DL205 Handheld Programmer Manual for details on using these AUX functions with the Handheld Programmer.

NOTE: *If the instructions are supported in both CPUs and the program size is within the limits of the D2-230, you can move a program between the two CPUs. However, the EEPROM installed in the Handheld Programmer must be the same size as (or larger than) the CPU being used. For example, you could not install a D2-240 EEPROM in the Handheld Programmer and download the program to a D2-230. Instead, if the program is within the size limits of the D2-230, use a D2-230 chip in the Handheld when you obtain the program from the D2-240.*

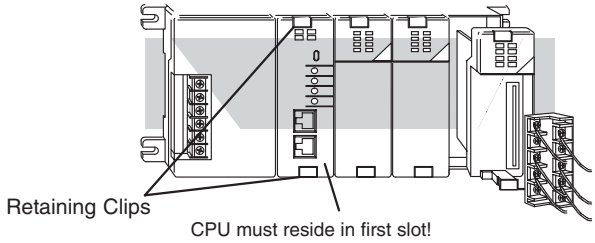
Installing the CPU

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

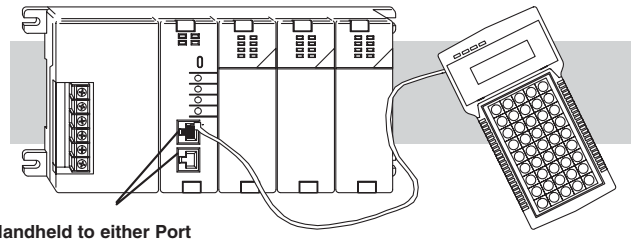
The CPU must be installed in the first slot in the base (closest to the power supply). You cannot install the CPU in any other slot. When inserting the CPU into the base, align the PC board with the grooves on the top and bottom of the base. Push the CPU straight into the base until it is firmly seated in the backplane connector. Use the retaining clips to secure the CPU to the base.

WARNING: To minimize the risk of electrical shock, personal injury, or equipment damage, always disconnect the system power before installing or removing any system component.

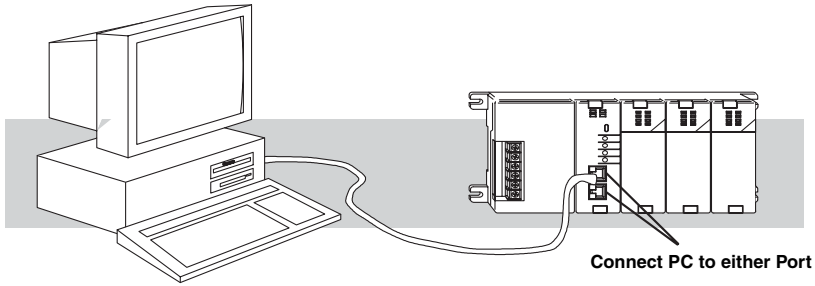
Connecting the Programming Devices

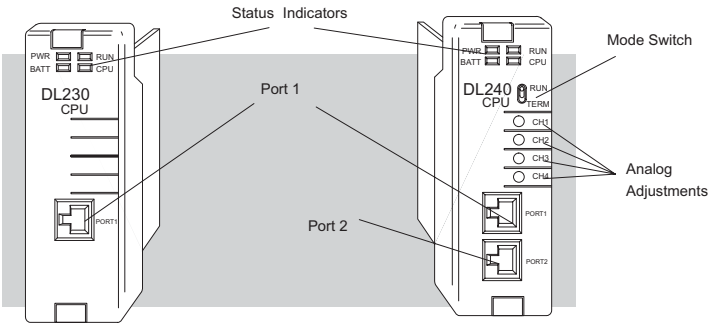


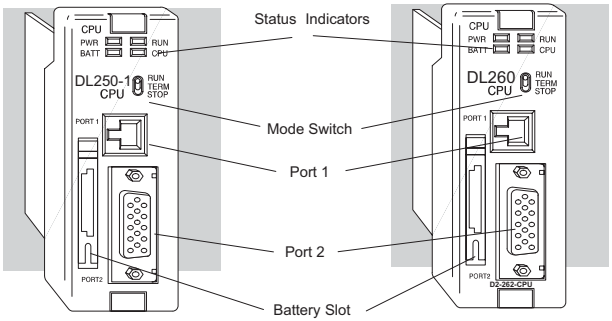
The handheld programmer is connected to the CPU with a Handheld Programmer cable. You can connect the Handheld Programmer to either port on a D2-240 CPU. The Handheld Programmer is shipped with a cable. The cable is approximately 6.5 ft (200cm).



If you are using a Personal Computer with the *Direct*SOFT programming package, you can use either the top or bottom port.







NOTE: D2-260 and D2-262 CPUs have the same faceplate features.

CPU Setup Information

Even if you have years of experience using PLCs, there are a few tasks you need to do before you can start entering programs. This section includes some basic tasks, such as changing the CPU mode, but it also includes some tasks that you may never have to use. Here's a brief list of the items that are discussed:

- Using auxiliary functions
- Clearing the program (and other memory areas)
- How to initialize system memory
- Setting retentive memory ranges

The following paragraphs provide the setup information necessary to ready the CPU for programming, including set-up instructions for either type of programming device you are using. The D2–HPP Handheld Programmer Manual provides the Handheld keystrokes required to perform all of these operations. The *Direct*SOFT Manual provides a description of the menus and keystrokes required to perform the setup procedures via *Direct*SOFT.

Status Indicators

The status indicator LEDs on the CPU front panels have specific functions that can help in programming and troubleshooting.

Indicator	Status	Meaning
PWR	ON	Power good
	OFF	Power failure
RUN	ON	CPU is in Run Mode
	OFF	CPU is in Stop or Program Mode
	Blinking	CPU is in Firmware Upgrade Mode
CPU	ON	CPU self diagnostics error
	OFF	CPU self diagnostics good
BATT	ON	Low battery voltage (only with System Memory bit B7633.12 set)
	OFF	CPU battery voltage is good or disabled

Mode Switch Functions

The mode switch on the D2-240, D2-250–1, D2-260 and D2-262 CPUs provides positions for enabling and disabling program changes in the CPU. Unless the mode switch is in the TERM position, RUN and STOP mode changes will not be allowed by any interface device, (Handheld Programmer, *Direct*SOFT programing package or operator interface). Programs may be viewed or monitored but no changes may be made. If the switch is in the TERM position and no program password is in effect, all operating modes as well as program access will be allowed through the connected programming or monitoring device.

The CPU mode can be changed in two ways:

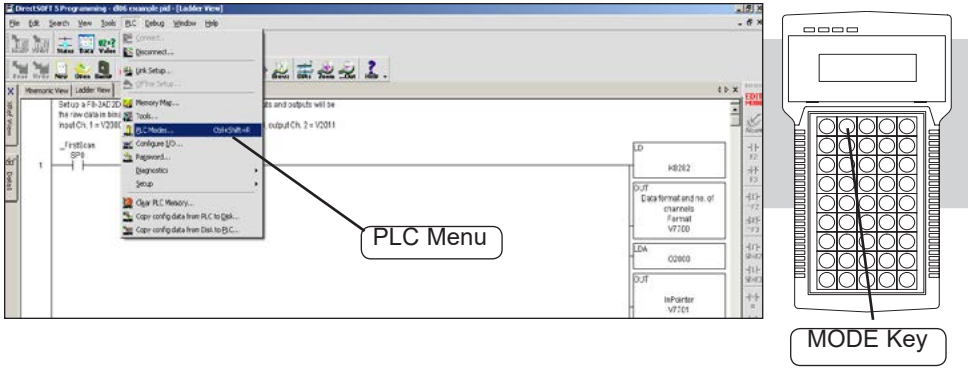
- Use the CPU mode switch to select the operating mode.
- Place the CPU mode switch in the TERM position and use a programming device to change operating modes. In this position, you can change between Run and Program modes.

NOTE: *If the PLC is switched to the RUN Mode without a program in the CPU, the CPU will produce a FATAL ERROR which can be cleared by cycling the power to the PLC.*

Mode Switch Position	CPU Action
RUN (Run Program)	CPU is forced into the RUN mode if no errors are encountered. No changes are allowed by the attached programming/monitoring device.
TERM (Terminal)	RUN, PROGRAM and the TEST modes are available. Mode and program changes are allowed by the programming/monitoring device.
STOP (D2-250–1, D2-260 and D2-262 only Stop Program)	CPU is forced into the STOP mode. No changes are allowed by the programming/monitoring device.

Changing Modes in the DL205 PLC

The CPU mode can be changed in two ways: you can use the CPU mode switch to select the operating mode, or you can place the mode switch in the TERM position and use a programming device to change operating modes. With the switch in this position, the CPU can be changed between Run and Program modes. You can use either *Direct*SOFT or the Handheld Programmer to change the CPU mode of operation. With *Direct*SOFT use the PLC menu option **PLC > Mode** or use the **Mode** button located on the Online toolbar. With the Handheld Programmer, use the MODE key.



Mode of Operation at Power Up

The DL205 CPUs will normally power up in the mode that it was in just prior to the power interruption. For example, if the CPU was in Program Mode when the power was disconnected, the CPU will power up in Program Mode (see warning note below).

WARNING: Once the super capacitor has discharged, the system memory may not retain the previous mode of operation. When this occurs, the PLC can power-up in either Run or Program Mode if the mode switch is in the term position. There is no way to determine which mode will be entered as the startup mode. Failure to adhere to this warning greatly increases the risk of unexpected equipment startup. For a D2-260, the super capacitor hold time is 15.9 hours. For a D2-262, the super capacitor hold time is 1.9 hours.

The mode in which the CPU will power up in is also determined by the state of System Memory bit B7633.13. If the bit is set and the Mode Switch is in the TERM position, the CPU will power-up in RUN mode. If B7633.13 is not set with the Mode Switch in TERM position, then the CPU will power up in the state it was in when it was powered down.

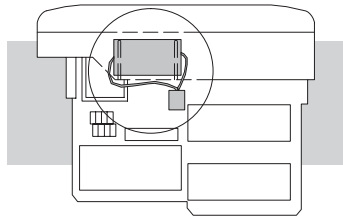
Using Battery Backup

An optional lithium battery is available to maintain the system RAM retentive memory when the DL205 system is without external power. Typical CPU battery life is five years, which includes PLC runtime and normal shut-down periods. However, consider installing a fresh battery if your battery has not been changed recently and the system will be shut down for a period of more than ten days.

NOTE: *Before installing or replacing your CPU battery, back up your V-memory and system parameters. You can do this by using* ***Direct****SOFT to save the program, V-memory, and system parameters to your personal computer hard-drive or a USB drive.*
As a reminder, the super capacitor hold time for a D2-260 is 15.9 hours. The super capacitor hold time for a D2-262 is 1.9 hours.

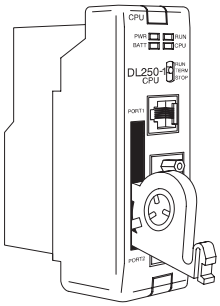
To install the D2–BAT CPU battery in D2-230 or D2-240 CPUs:

1. Gently push the battery connector onto the circuit board connector (Shown at right).
2. Push the battery into the retaining clip. Don't use excessive force. You may break the retaining clip.
3. Make a note of the date the battery was installed.



D2-230 and D2-240

D2-250-1, D2-260 and D2-262



To install the D2–BAT–1 CPU battery in the D2-250–1, D2-260 and D2-262 CPUs: (#CR2354)

1. Press the retaining clip on the battery door down and swing the battery door open.
2. Place the battery into the coin–type slot with the +, or larger, side out.
3. Close the battery door making sure that it locks securely in place
4. Make a note of the date the battery was installed.

WARNING: Do not attempt to recharge the battery or dispose of an old battery by fire. The battery may explode or release hazardous materials.

Battery Backup

The battery backup is available immediately after the battery has been installed in the DL205 CPUs. ***The battery low (BATT) indicator will turn on if the battery is less than 2.5VDC*** (refer to the Status Indicator table on page 3-12). Special Relay 43 (SP43) will also be activated. The low battery indication is enabled by setting bit 12 of V7633 (B7633.12). If the low-battery feature is not desired, do not set bit V7633.12.

The super capacitor will retain memory IF it is configured as retentive regardless of the state of B7633.12. The battery will be the same, but for a much longer time.

Auxiliary Functions

Many CPU set-up tasks involve the use of Auxiliary (AUX) Functions. The AUX Functions perform many different operations, including clearing ladder memory, displaying the scan time, copying programs to EEPROM in the Handheld Programmer, etc. They are divided into categories that affect different system parameters. Appendix A provides a description of the AUX functions.

You can access the AUX Functions from *Direct*SOFT or from the DL205 Handheld Programmer. The manuals for those products provide step-by-step procedures for accessing the AUX Functions. Some of these AUX Functions are designed specifically for the Handheld Programmer setup, so they will not be needed (or available) with the *Direct*SOFT package. The following table shows a list of the Auxiliary functions for the different CPUs and the Handheld Programmer.

NOTE: *The Handheld Programmer may have additional AUX functions that are not supported with the DL205 CPUs.*

AUX Function and Description		230	240	250–1	260/ 262
AUX 2* — RLL Operations					
21	Check Program	✓	✓	✓	✓
22	Change Reference	✓	✓	✓	✓
23	Clear Ladder Range	✓	✓	✓	✓
24	Clear All Ladders	✓	✓	✓	✓
AUX 3* — V-Memory Operations					
31	Clear V Memory	✓	✓	✓	✓
AUX 4* — I/O Configuration					
41	Show I/O Configuration	✓	✓	✓	✓
42	I/O Diagnostics	✓	✓	✓	✓
44	Power-up I/O Configuration Check	✓	✓	✓	✓
45	Select Configuration	✓	✓	✓	✓
46	Configure I/O	X	X	✓	✓
AUX 5* — CPU Configuration					
51	Modify Program Name	✓	✓	✓	✓
52	Display /Change Calendar	X	✓	✓	✓
53	Display Scan Time	✓	✓	✓	✓
54	Initialize Scratchpad	✓	✓	✓	✓
55	Set Watchdog Timer	✓	✓	✓	✓
56	Set CPU Network Address	X	✓	✓	✓
57	Set Retentive Ranges	✓	✓	✓	✓
58	Test Operations	✓	✓	✓	✓
59	Bit Override	X	✓	✓	✓
5B	Counter Interface Config.	✓	✓	✓	✓
5C	Display Error History	X	✓	✓	✓

AUX Function and Description		230	240	250–1	260/ 262	HPP
AUX 6* — Handheld Programmer Configuration						
61	Show Revision Numbers	✓	✓	✓	✓	–
62	Beeper On / Off	X	X	X	X	✓
65	Run Self Diagnostics	X	X	X	X	✓
AUX 7* — EEPROM Operations						
71	Copy CPU memory to HPP EEPROM	X	X	X	X	✓
72	Write HPP EEPROM to CPU	X	X	X	X	✓
73	Compare CPU to HPP EEPROM	X	X	X	X	✓
74	Blank Check (HPP EEPROM)	X	X	X	X	✓
75	Erase HPP EEPROM	X	X	X	X	✓
76	Show EEPROM Type (CPU and HPP)	X	X	X	X	✓
AUX 8* — Password Operations						
81	Modify Password	✓	✓	✓	✓	–
82	Unlock CPU	✓	✓	✓	✓	–
83	Lock CPU	✓	✓	✓	✓	–

✓ *Supported*
X *Not Supported*
\- *Not Applicable*

Clearing an Existing Program

Before you enter a new program, you should always clear ladder memory. You can use AUX Function 24 to clear the complete program.

You can also use other AUX functions to clear other memory areas.

- AUX 23 — Clear Ladder Range
- AUX 24 — Clear all Ladders
- AUX 31 — Clear V-Memory

Initializing System Memory

The DL205 CPUs maintain system parameters in a memory area often referred to as the "scratchpad." In some cases, you may make changes to the system setup that will be stored in system memory. For example, if you specify a range of Control Relays (CRs) as retentive, these changes are stored. AUX 54 resets the system memory to the default values.

WARNING: You may never have to use this feature unless you want to clear any set-up information that is stored in system memory. Usually, you will only need to initialize the system memory if you are changing programs and the old program required a special system setup. You can usually change from program to program without ever initializing system memory. Remember, this AUX function will reset all system memory. If you have set special parameters such as retentive ranges, etc., they will be erased when AUX 54 is used. Make sure that you have considered all ramifications of this operation before you select it.

Setting the Clock and Calendar

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The D2-240, D2-250–1, D2-260 and D2-262 also have a Clock/Calendar that can be used for many purposes. If you need to use this feature, AUX functions are available that allow you to set the date and time. For example, you would use AUX 52, Display/Change Calendar to set the time and date with the Handheld Programmer. With ***Direct***SOFT you would use the PLC set-up menu options using K–Sequence protocol only.

The CPU uses the following format to display the date and time.

Handheld Programmer Display

```
23:08:17  08/02/20
```

- Date — Year, Month, Date, Day of week
  (0 – 6, Sunday through Saturday)
- Time — 24-hour format, Hours, Minutes, Seconds

You can use the AUX function to change any component of the date or time. However, the CPU will not automatically correct any discrepancy between the date and the day of the week. For example, if you change the date to the 15th of the month and the 15th is on a Thursday, you will also have to change the day of the week (unless the CPU already shows the date as Thursday). The day of the week can only be set using the Handheld Programmer.

Setting the CPU Network Address

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The D2-240, D2-250–1, D2-260 and D2-262 CPUs have built in *Direct*Net ports. You can use the Handheld Programmer to set the network address for the port and the port communication parameters. The default settings are:

- Station Address 1
- Hex Mode
- Odd Parity
- 9600 Baud

The *Direct*Net Manual provides additional information about choosing the communication settings for network operation.

Setting Retentive Memory Ranges

The DL205 CPUs provide certain ranges of retentive memory by default. The default ranges are suitable for many applications, but you can change them if your application requires additional retentive ranges or no retentive ranges at all. The default settings are:

Memory Area	D2-230		D2-240		D2-250–1		D2-260/D2-262	
	Default Range	Avail. Range	Default Range	Avail. Range	Default Range	Avail. Range	Default Range	Avail. Range
Control Relays	C300 – C377	C0 – C377	C300 – C377	C0 – C377	C1000 – C1777	C0 – C1777	C1000 – C3777	C0 – C3777
V-Memory	V2000 – V7777	V0 – V7777	V2000 – V7777	V0 – V7777	V1400 – V3777	V0 – V17777	V400 – V37777	V0 – V37777
Timers	None by default	T0 – T77	None by default	T0 – T177	None by default	T0 – T377	None by default	T0 – T377
Counters	CT0 – CT77	CT0 – CT77	CT0 – CT177	CT0 – CT177	CT0 – CT177	CT0 – CT177	CT0 – CT377	CT0 – CT377
Stages	None by default	S0 – S377	None by default	S0 – S777	None by default	S0 – S1777	None by default	S0 – S1777

You can use AUX 57 to set the retentive ranges. You can also use *Direct*SOFT menus to select the retentive ranges.

WARNING: The DL205 CPUs do not come with a battery. The super capacitor will retain the values in the event of a power loss, but only for a short period of time, depending on conditions. If the retentive ranges are important for your application, make sure you obtain the optional battery.

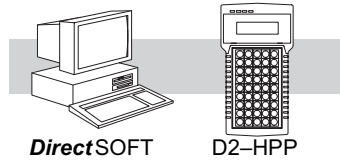
Using a Password

The DL205 CPUs allow you to use a password to help minimize the risk of unauthorized program and/or data changes. Once you enter a password you can "lock" the CPU against access. Once the CPU is locked you must enter the password before you can use a programming device to change any system parameters.

You can select an 8-digit numeric password. The CPUs are shipped from the factory with a password of 00000000. All zeros removes the password protection. If a password has been entered into the CPU, you cannot enter all zeros to remove it. Once you enter the correct password, you can change the password to all zeros to remove the password protection. For more information on passwords, see the appropriate appendix on auxiliary functions.

WARNING: Make sure you remember your password. If you forget your password you will not be able to access the CPU. The CPU must be returned to the factory to have the password (along with the ladder project) removed. It is the policy of AutomationDirect to require the memory of the PLC to be cleared along with the password.

You can use the D2–HPP Handheld Programmer or *Direct*SOFT to enter a password. The following diagram shows how you can enter a password with the Handheld Programmer.



Select AUX 81

CLR CLR I 8 B 1 AUX ENT

PASSWORD
00000000

Enter the new 8-digit password

X X • • • X ENT

PASSWORD
XXXXXXXX

Press CLR to clear the display

The CPU can be locked three ways once the password has been entered.

- If the CPU power is disconnected, the CPU will be automatically locked against access.
- If you enter the password with *Direct*SOFT, the CPU will be automatically locked against access when you exit *Direct*SOFT.
- Use AUX 83 to lock the CPU.

When you use *Direct*SOFT, you will be prompted for a password if the CPU has been locked. If you use the Handheld Programmer, you have to use AUX 82 to unlock the CPU. Once you enter AUX 82, you will be prompted to enter the password.

***NOTE**: The D2-240, D2-250–1, D2-260 and D2-262 CPUs offer multi–level passwords for even more password protection of the ladder program. This allows password protection while not locking the communication port to an operator interface. The multi-level password can be invoked by creating a password with an upper case "A" followed by seven numeric characters (e.g., A1234567).*

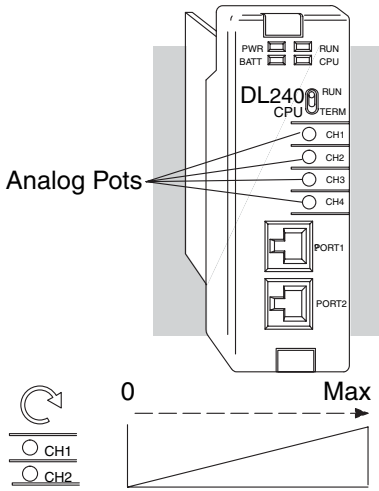
Setting the Analog Potentiometer Ranges

☒ 230
☑ 240
☒ 250-1
☒ 260
☒ 262

Four analog potentiometers (pots) are on the face plate of the D2-240 (DL240 CPU). These pots can be used to change timer constants, frequency of pulse train output, value for an analog output module, etc.

Each analog channel has corresponding V-memory locations for setting lower and upper limits for each analog channel.

To increase the value associated with the analog pot, turn the pot clockwise. To decrease the value, turn the pot counter clockwise. Turn clockwise to increase value.



The table below shows the V-memory locations used for each analog channel. These are the default locations for the analog pots.

	CH1	CH2	CH3	CH4
Analog Data	V3774	V3775	V3776	V3777
Analog Data Lower Limit	V7640	V7642	V7644	V7646
Analog Data Upper Limit	V7641	V7643	V7645	V7647

You can use the program logic to load the limits into these locations, or, you can use a programming device to load the values. The range for each limit is 0 – 9999.

These analog pots have a resolution of 256 pieces. Therefore, if the span between the upper and lower limits is less than or equal to 256, then you have better resolution or, more precise control.

Use the formula shown to determine the smallest amount of change that can be detected.

For example, a range of 100 – 600 would result in a resolution of 1.95. Therefore, the smallest increment would be 1.95 units. (The actual result depends on exactly how you are using the values in the control program).

$$\text{Resolution} = \frac{H - L}{256}$$

H = high limit of the range
L = low limit of the range

Example Calculations:

H = 600
L = 100

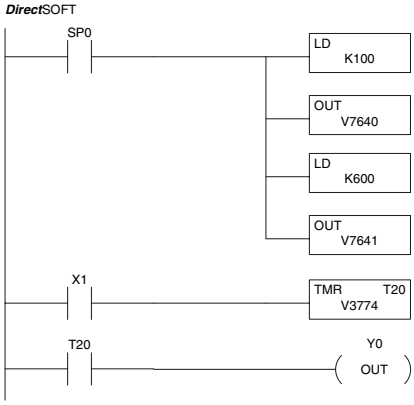
$$\text{Resolution} = \frac{600-100}{256}$$

$$\text{Resolution} = \frac{500}{256}$$

$$\text{Resolution} = 1.95$$

The following example shows how you could use these analog potentiometers to change the preset value for a timer. See Chapter 5 for details on how these instructions operate.

Program loads ranges into V-memory

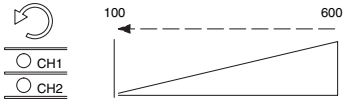


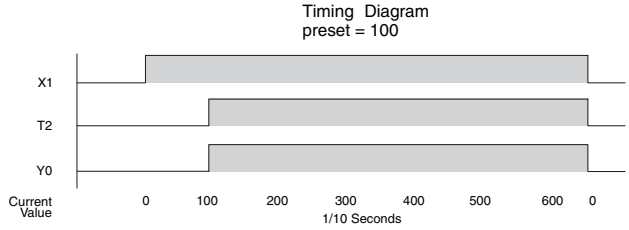
Load the lower limit (100) for the analog range on Ch1 into V7640.

Load the upper limit (600) for the analog range on Ch1 into V7641.

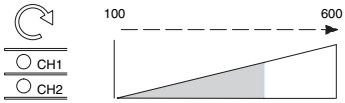
Use V3774 as the preset for the timer. This will allow you to quickly adjust the preset from 100 to 600 with the CH1 analog pot.

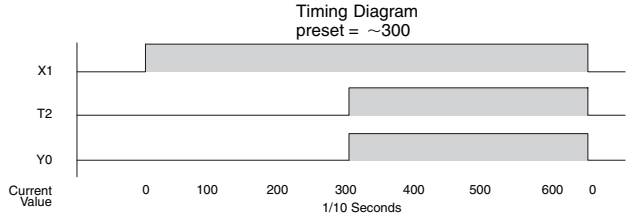
Turn all the way counter-clockwise to use lowest value





Turn clockwise to increase the timer preset.

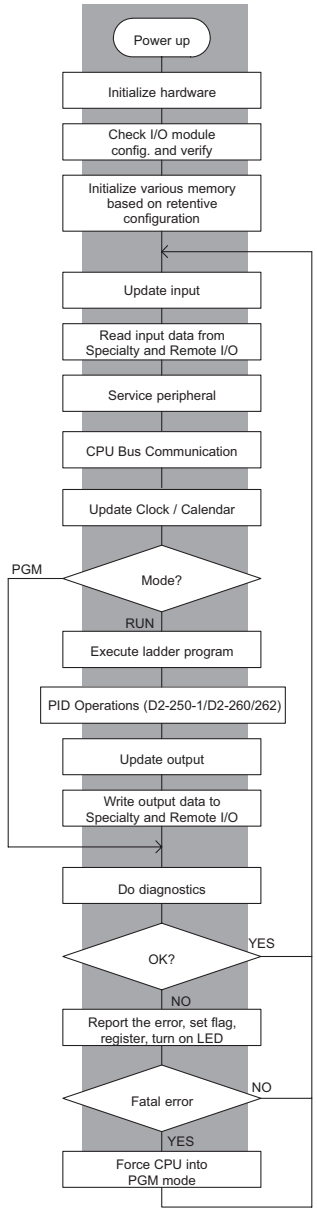




CPU Operation

Achieving the proper control for your equipment or process requires a good understanding of how DL205 CPUs control all aspects of system operation. The flowchart below shows the main tasks of the CPU operating system. In this section, we will investigate four aspects of CPU operation:

- CPU Operating System — The CPU manages all aspects of system control.
- CPU Operating Modes — The three primary modes of operation are Program Mode, Run Mode, and Test Mode.
- CPU Timing — The two important areas we discuss are the I/O response time and the CPU scan time.
- CPU Memory Map — The CPU's memory map shows the CPU addresses of various system resources, such as timers, counters, inputs, and outputs.



CPU Operating System

At power up, the CPU initializes the internal electronic hardware. Memory initialization starts with examining the retentive memory settings. In general, the contents of retentive memory are preserved, and non-retentive memory is initialized to zero (unless otherwise specified).

After the one-time power-up tasks, the CPU begins the cyclical scan activity. The flowchart to the right shows how the tasks differ based on the CPU mode and the existence of any errors. The "scan time" is defined as the average time around the task loop. Note that the CPU is always reading the inputs, even during program mode. This allows programming tools to monitor input status at any time.

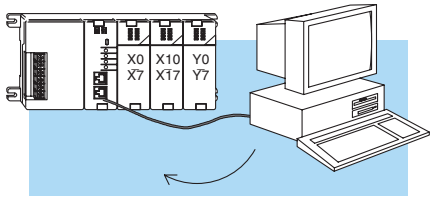
The outputs are only updated in Run mode. In Program mode, they are in the off state.

In Run Mode, the CPU executes the user ladder program. Immediately afterwards, any PID loops which are configured are executed (D2-250-1, D2-260 and D2-262). Then the CPU writes the output results of these two tasks to the appropriate output points.

Error detection has two levels: Non-fatal and fatal. Non-fatal errors are reported, but the CPU remains in its current mode. If a fatal error occurs, the CPU is forced into program mode and the outputs go off.

Program Mode Operation

In Program Mode the CPU does not execute the application program or update the output modules. The primary use for Program Mode is to enter or change an application program. You also use the program mode to set up CPU parameters, such as the network address, retentive memory areas, etc.



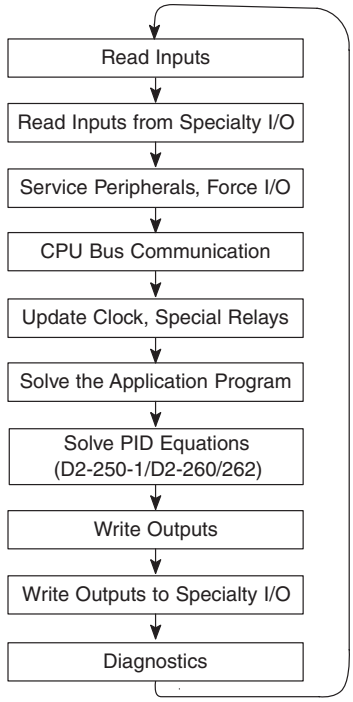
Download Program

You can use the mode switch on the D2-250–1, D2-260 and D2-262 CPUs to select Program Mode operation. Or, with the switch in TERM position, you can use a programming device such as the Handheld Programmer to place the CPU in Program Mode.

Run Mode Operation

In Run Mode, the CPU executes the application program, does PID calculations for configured PID loops (D2-250-1, D2-260 and D2-262), and updates the I/O system. You can perform many operations during Run Mode. Some of these include:

- Monitor and change I/O point status
- Update timer/counter preset values
- Update Variable memory locations



Run Mode operation can be divided into several key areas. It is very important you understand how each of these areas of execution can affect the results of your application program solutions.

You can use the mode switch to select Run Mode operation (D2-240, D2-250–1, D2-260 and D2-262). Or, with the mode switch in TERM position, you can use a programming device, such as the Handheld Programmer, to place the CPU in Run Mode.

You can also edit the program during Run Mode. The Run Mode Edits are not "bumpless." Instead, the CPU maintains the outputs in their last state while it accepts the new program information. If an error is found in the new program, then the CPU will turn all the outputs off and enter the Program Mode.

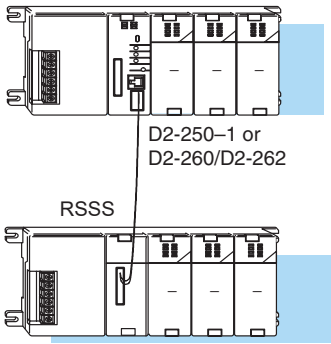
WARNING: Only authorized personnel fully familiar with all aspects of the application should make changes to the program. Changes during Run Mode become effective immediately. Make sure you thoroughly consider the impact of any changes to minimize the risk of personal injury or damage to equipment.

Read Inputs

The CPU reads the status of all inputs, then stores it in the image register. Input image register locations are designated with an X followed by a memory location. Image register data is used by the CPU when it solves the application program. Of course, an input may change after the CPU has read the inputs. Generally, the CPU scan time is measured in milliseconds. If you have an application that cannot wait until the next I/O update, you can use Immediate Instructions. These do not use the status of the input image register to solve the application program. The Immediate instructions immediately read the input status directly from I/O modules. However, this lengthens the program scan since the CPU has to read the I/O point status again. A complete list of the Immediate instructions is included in Chapter 5.

Read Inputs from Specialty and Remote I/O

After the CPU reads the inputs from the input modules, it reads any input point data from any Specialty modules that are installed, such as Counter Interface modules, etc. This is also the portion of the scan that reads the input status from Remote I/O bases.



***NOTE**: It may appear the Remote I/O point status is updated every scan. This is not quite true. The CPU will receive information from the Remote I/O Master module every scan, but the Remote Master may not have received an update from all the Remote Slaves. Remember, the Remote I/O link is managed by the Remote Master, not the CPU.*

Service Peripherals and Force I/O

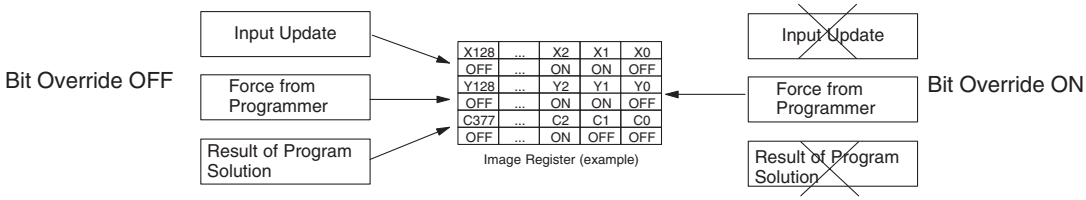
After the CPU reads the inputs from the input modules, it reads any attached peripheral devices. This is primarily a communications service for any attached devices. For example, it would read a programming device to see if any input, output, or other memory type status needs to be modified. Two basic types of forcing are available with the DL205 CPUs.

***NOTE**: **Direct**Net protocol does not support bit operations.*

- Forcing from a peripheral – not a permanent force, good only for one scan
- Bit Override (D2-240, D2-250–1, D2-260 and D2-262) – holds the I/O point (or other bit) in the current state. Valid bits are X, Y, C, T, CT, and S. These memory types are discussed in more detail later in this chapter.

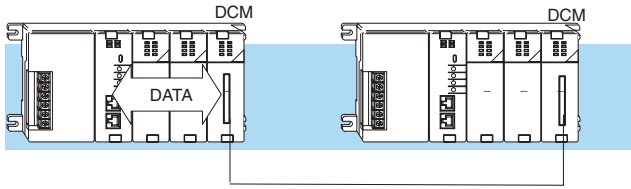
Regular Forcing — This type of forcing can temporarily change the status of a discrete bit. For example, you may want to force an input on, even though it is really off. This allows you to change the point status that was stored in the image register. This value will be valid until the image register location is written to during the next scan. This is primarily useful during testing situations when you need to force a bit on to trigger another event.

Bit Override — (D2-240, D2-250–1, D2-260 and D2-262) Bit override can be enabled on a point-by-point basis by using AUX 59 from the Handheld Programmer or, by a menu option from within ***Direct***SOFT. Bit override basically disables any changes to the discrete point by the CPU. For example, if you enable bit override for X1, and X1 is off at the time, then the CPU will not change the state of X1. This means that even if X1 comes on, the CPU will not acknowledge the change. So, if you used X1 in the program, it would always be evaluated as "off" in this case. Of course, if X1 was on when the bit override was enabled, then X1 would always be evaluated as "on." There is an advantage available when you use the bit override feature. The regular forcing is not disabled because the bit override is enabled. For example, if you enabled the Bit Override for Y0 and it was off at the time, then the CPU would not change the state of Y0. However, you can still use a programming device to change the status. Now, if you use the programming device to force Y0 on, it will remain on and the CPU will not change the state of Y0. If you then force Y0 off, the CPU will maintain Y0 as off. The CPU will never update the point with the results from the application program or from the I/O update until the bit override is removed. The following diagram shows a brief overview of the bit override feature. Notice the CPU does not update the Image Register when bit override is enabled.



CPU Bus Communication

Specialty Modules, such as the Data Communications Module, can transfer data to and from the CPU over the CPU bus on the backplane. This data is more than standard I/O point status. This type of communications can only occur on the CPU (local) base. A portion of the execution cycle is used to communicate with these modules. The CPU performs both read and write requests during this segment.



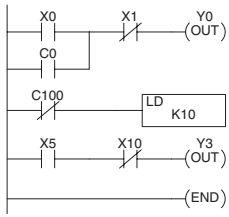
Update Clock, Special Relays and Special Registers

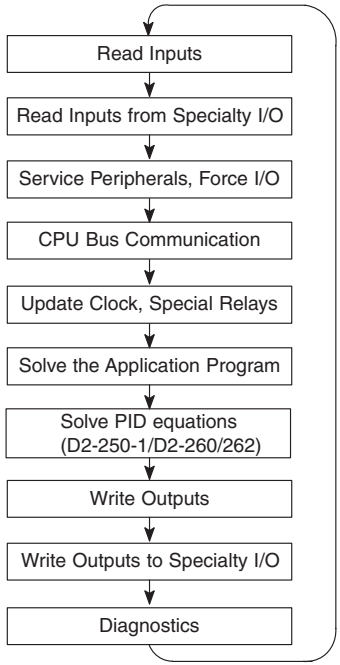
The D2-240, D2-250–1, D2-260 and D2-262 CPUs have an internal real-time clock and calendar timer which are accessible to the application program. Special V-memory locations hold this information. This portion of the execution cycle makes sure these locations get updated on every scan. Several different Special Relays, such as diagnostic relays, etc., are also updated during this segment.

Solve Application Program

The CPU evaluates each instruction in the application program during this segment of the scan cycle. The instructions define the relationship between input conditions and the system outputs.

The CPU begins with the first rung of the ladder program, evaluating it from left to right and from top to bottom. It continues, rung by rung, until it encounters the END coil instruction. At that point, a new image for the outputs is complete.





The internal control relays (C), the stages (S), and the variable memory (V) are also updated in this segment.

You may recall the CPU may have obtained and stored forcing information when it serviced the peripheral devices. If any I/O points or memory data have been forced, the output image register also contains this information.

***NOTE**: If an output point was used in the application program, the results of the program solution will overwrite any forcing information that was stored. For example, if Y0 was forced on by the programming device, and a rung containing Y0 was evaluated such that Y0 should be turned off, then the output image register will show that Y0 should be off. Of course, you can force output points that are not used in the application program. In this case, the point remains forced because there is no solution that results from the application program execution.*

Solve PID Loop Equations

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The D2-260 and D2-262 CPUs can process up to 16 PID loops and the D2-250–1 can process up to 4 PID loops. The loop calculations are run as a separate task from the ladder program execution, immediately following it. Only loops that have been configured are calculated, and then only according to a built-in loop scheduler. The sample time (calculation interval) of each loop is programmable. Please refer to Chapter 8, PID Loop Operation, for more on the effects of PID loop calculation on the overall CPU scan time.

Write Outputs

Once the application program has solved the instruction logic and constructed the output image register, the CPU writes the contents of the output image register to the corresponding output points located in the local CPU base or the local expansion bases. Remember, the CPU also made sure any forcing operation changes were stored in the output image register, so the forced points get updated with the status specified earlier.

Write Outputs to Specialty and Remote I/O

After the CPU updates the outputs in the local and expansion bases, it sends the output point information that is required by any Specialty modules that are installed. For example, this is the portion of the scan that writes the output status from the image register to the Remote I/O racks.

***NOTE**: It may appear the Remote I/O point status is updated every scan. This is not quite true. The CPU will send the information to the Remote I/O Master module every scan, but the Remote Master will update the actual remote modules during the next communication sequence between the master and slave modules. Remember, the Remote I/O link communication is managed by the Remote Master, not the CPU.*

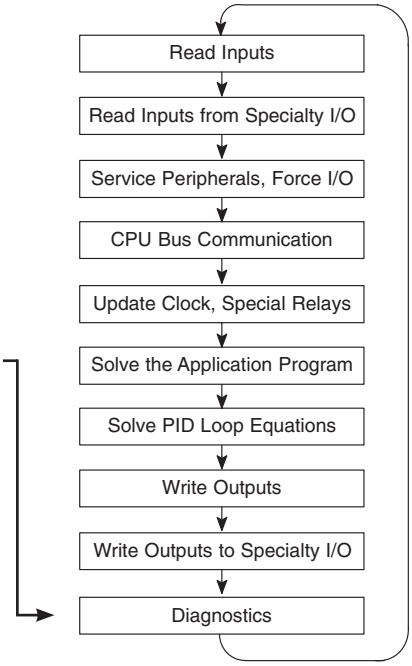
Diagnostics

During this part of the scan, the CPU performs all system diagnostics and other tasks, such as:

- Calculating the scan time
- Updating special relays
- Resetting the watchdog timer

DL205 CPUs automatically detect and report many different error conditions. Appendix B contains a listing of the various error codes available with the DL205 system.

One of the more important ***Diagnostics*** tasks is the scan time calculation and watchdog timer control. DL205 CPUs have a "watchdog" timer that stores the maximum time allowed for the CPU to complete the solve application segment of the scan cycle. The default value set from the factory is 200ms. If this time is exceeded the CPU will enter the Program Mode, turn off all outputs, and report the error. For example, the Handheld Programmer displays "E003 S/W TIMEOUT" when the scan overrun occurs.



You can use AUX 53 to view the minimum, maximum, and current scan time. Use AUX 55 to increase or decrease the watchdog timer value. There is also an RSTWT instruction that can be used in the application program to reset the watch dog timer during the CPU scan.

I/O Response Time

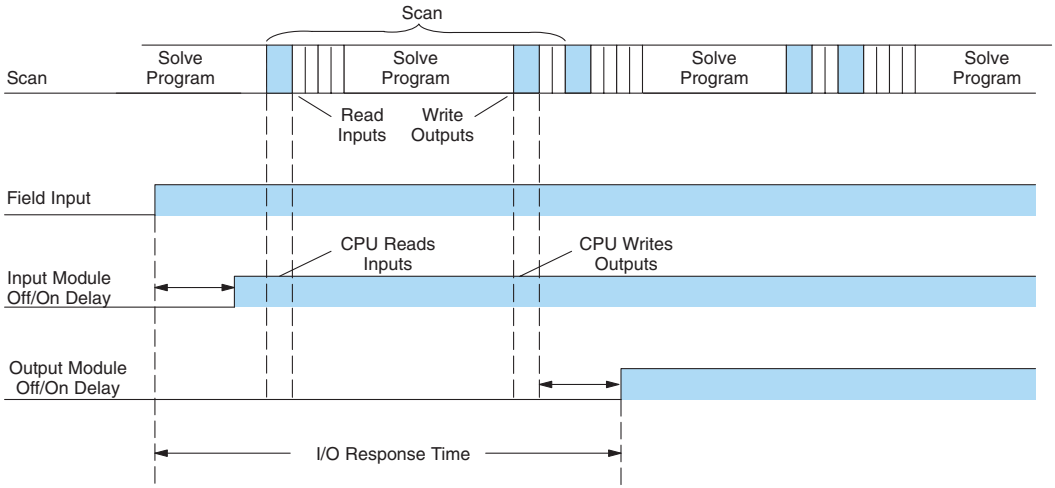
Is Timing Important for Your Application?

I/O response time is the amount of time required for the control system to sense a change in an input point and update a corresponding output point. In the majority of applications, the CPU performs this task practically instantaneously. However, some applications do require extremely fast update times. Four things can affect the I/O response time:

- The point in the scan period when the field input changes states
- Input module Off to On delay time
- CPU scan time
- Output module Off to On delay time

Normal Minimum I/O Response

The I/O response time is shortest when the module senses the input change before the Read Inputs portion of the execution cycle. In this case the input status is read, the application program is solved, and the output point gets updated. The following diagram shows an example of the timing for this situation.



In this case, you can calculate the response time by simply adding the following items:

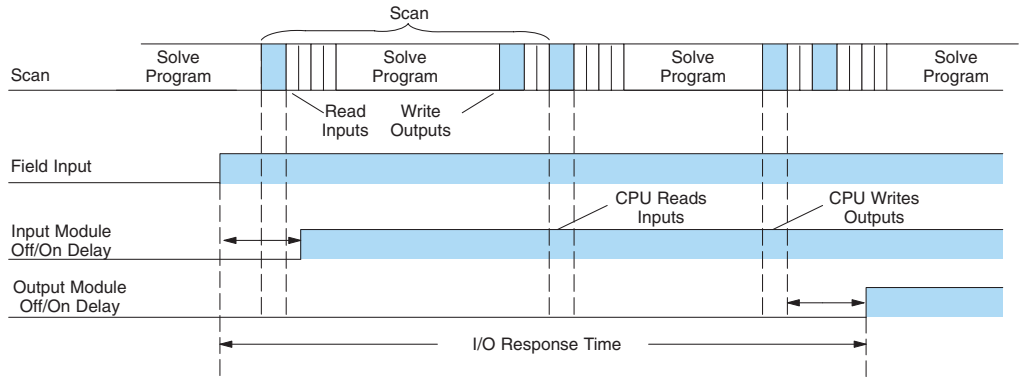
Input Delay + Scan Time + Output Delay = Response Time

Normal Maximum I/O Response

The I/O response time is longest when the module senses the input change after the Read Inputs portion of the execution cycle. In this case the new input status does not get read until the following scan. The following diagram shows an example of the timing for this situation.

In this case, you can calculate the response time by simply adding the following items:

Input Delay +(2 x Scan Time) + Output Delay = Response Time

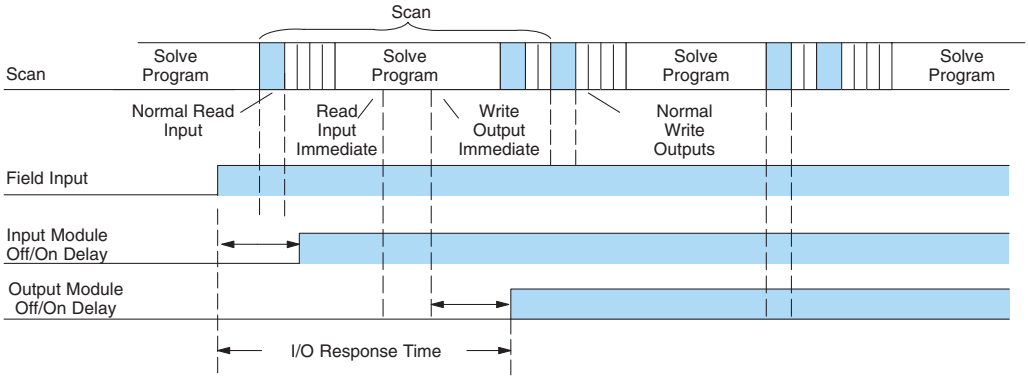


Improving Response Time

You can do a few things to help improve throughput.

- Choose instructions with faster execution times
- Use immediate I/O instructions (which update the I/O points during the ladder program execution segment)
- Choose modules that have faster response times

Immediate I/O instructions are probably the most useful technique. The following example shows immediate input and output instructions and their effect.



In this case, you can calculate the response time by simply adding the following items:

Input Delay + Instruction Execution Time + Output Delay = Response Time

The instruction execution time is calculated by adding the time for the immediate input instruction, the immediate output instruction, and all instructions in between.

***NOTE**: When the immediate instruction reads the current status from a module, it uses the results to solve that one instruction without updating the image register. Therefore, any regular instructions that follow will still use image register values. Any immediate instructions that follow will access the module again to update the status.*

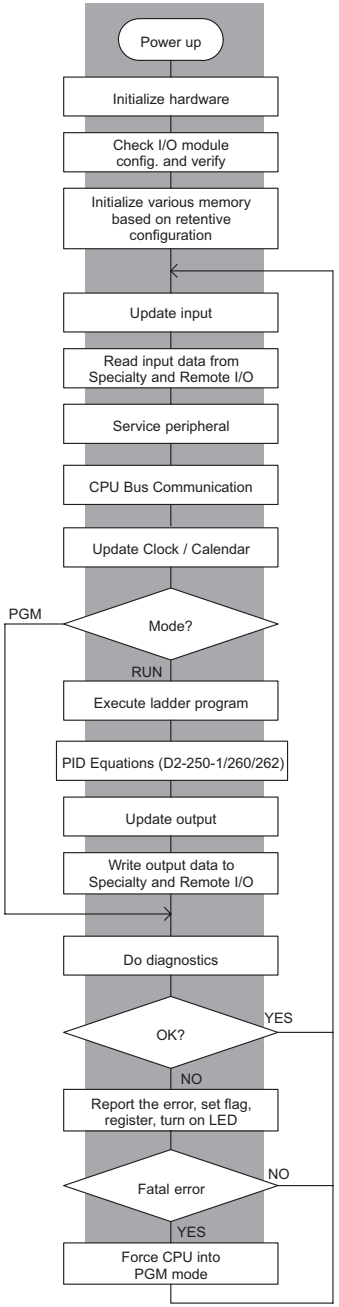
CPU Scan Time Considerations

The scan time covers all the cyclical tasks that the operating system performs. You can use *Direct*SOFT or the Handheld Programmer to display the minimum, maximum, and current scan times that have occurred since the previous Program Mode to Run Mode transition. This information can be very important when evaluating system performance.

As shown previously, there are several segments that make up the scan cycle. Each of these segments requires a certain amount of time to complete. Of all the segments, the only one you really have the most control over is the amount of time it takes to execute the application program. This is because different instructions take different amounts of time to execute. So, if you think you need a faster scan, then you can try to choose faster instructions.

Your choice of I/O modules and system configuration, such as expansion or remote I/O, can also affect the scan time; however, the application usually dictates them.

For example, if you need to count pulses at high rates of speed, then you will probably have to use a High-Speed Counter module. Also, if you have I/O points that need to be located several hundred feet from the CPU, then you need remote I/O because it is much faster and cheaper to install a single remote I/O cable than it is to run all those signal wires for each individual I/O point. The following paragraphs provide some general information on how much time some of the segments can require.



Initialization Process

The CPU performs an initialization task once the system power is on. The initialization task is performed once at power up, so it does not affect the scan time for the application program.

Reading Inputs

Initialization	D2-230	D2-240	D2-250–1	D2-260/D2-262
Minimum Time	1.6 seconds	1.0 seconds	1.2 seconds	1.2 seconds
Maximum Time	3.6 seconds	2.0 seconds	2.7 seconds(w/ 2 exp. bases)	3.7 seconds (w/ 4 exp. bases)

The time required to read the input status for the input modules depends on which CPU you are using and the number of input points in the base. The following table shows typical update times required by the CPU.

Timing Factors	D2-230	D2-240	D2-250–1	D2-260/D2-262
Overhead	64.0 µs	32.0 µs	12.6 µs	12.6 µs
Per input point	6.0 µs	12.3 µs	2.5 µs	2.5 µs

For example, the time required for a D2-240 to read two 8-point input modules would be calculated as follows, where NI is the total number of input points:

Formula

Time = 32µs + (12.3 x NI)

Example

Time = 32µs + (12.3 x 16)

Time = 228.8 µs

***NOTE**: This information provides the amount of time the CPU spends reading the input status from the modules. Don't confuse this with the I/O response time that was discussed earlier.*

Reading Inputs from Specialty I/O

During this portion of the cycle the CPU reads any input points associated with the following:

- Remote I/O
- Specialty Modules (such as High-Speed Counter, etc)

The time required to read any input status from these modules depends on which CPU you are using, the number of modules, and the number of input points.

Remote Module	D2-230	D2-240	D2-250–1	D2-260/D2-262
Overhead	N/A	6.0 µs	1.82 µs	1.82 µs
Per module (with inputs)	N/A	67.0 µs	17.9 µs	17.9 µs
Per input point	N/A	40.0 µs	2.0 µs	2.0 µs

For example, the time required for a D2-240 to read two 8-point input modules (located in a Remote base) would be calculated as follows, where NM is the number of modules and NI is the total number of input points:

Remote I/O

Formula

Time = 6µs + (67µs x NM) + (40µs x NI)

Example

Time = 6µs + (67µs x 2) + (40µs x 16)

Time = 780µs

Service Peripherals

Communication requests can occur at any time during the scan, but the CPU only "logs" the requests for service until the Service Peripherals portion of the scan. The CPU does not spend any time on this if there are no peripherals connected.

To Log Request (anytime)		D2-230	D2-240	D2-250–1	D2-260/D2-262
Nothing Connected	Min. & Max.	0 µs	0 µs	0 µs	0 µs
Port 1	Send Min. / Max.	22/28 µs	23/26 µs	3.2/9.2 µs	3.2/9.2 µs
	Rec. Min. / Max.	24/58 µs	52/70 µs	25.0/35.0 µs	25.0/35.0 µs
Port 2	Send Min. / Max.	N/A	26/30 µs	3.6/11.5 µs	3.6/11.5 µs
	Rec. Min. / Max.	N/A	60/75 µs	35.0/44.0 µs	35.0/44.0 µs

During the Service Peripherals portion of the scan, the CPU analyzes the communications request and responds as appropriate. The amount of time required to service the peripherals depends on the content of the request.

To Service Request	D2-230	D22-240	D2-250–1	D2-260/D2-262
Minimum	260µs	250µs	8µs	8µs
Run Mode Max.	30ms	20ms	410µs	410µs
Program Mode Max.	3.5 Seconds	4 Seconds	2 Seconds	3.7 Seconds

CPU Bus Communication

Some specialty modules can also communicate directly with the CPU via the CPU bus. During this portion of the cycle the CPU completes any CPU bus communications. The actual time required depends on the type of modules installed and the type of request being processed.

***NOTE**: Some specialty modules can have a considerable impact on the CPU scan time. If timing is critical in your application, consult the module documentation for any information concerning the impact on the scan time.*

Update Clock/Calendar, Special Relays, Special Registers

The clock, calendar, and special relays are updated and loaded into special V-memory locations during this time. This update is performed during both Run and Program Modes.

Modes		D2-230	D2-240	D2-250–1	D2-260/D2-262
Program Mode	Minimum	8.0 µs fixed	35.0 µs	11.0 µs	11.0 µs
	Maximum	8.0 µs fixed	48.0 µs	11.0 µs	11.0 µs
Run Mode	Minimum	20.0 µs	60.0 µs	19.0 µs	19.0 µs
	Maximum	26.0 µs	85.0 µs	26.0 µs	26.0 µs

Writing Outputs

The time required to write the output status for the local and expansion I/O modules depends on which CPU you are using and the number of output points in the base. The following table shows typical update times required by the CPU.

Timing Factors	D2-230	D2-240	D2-250–1	D2-260/D2-262
Overhead	66.0 µs	33.0 µs	28.1 µs	28.1 µs
Per output point	8.5 µs	14.6 µs	3.0 µs	3.0 µs

For example, the time required for a D2-240 to write data for two 8-point output modules would be calculated as follows (where NO is the total number of output points):

Formula

Time = 33 + (NO x 14.6 µs)

Example

Time = 33 + (16 x 14.6 µs)

Time = 266.6 µs

Writing Outputs to Specialty I/O

During this portion of the cycle the CPU writes any output points associated with the following.

- Remote I/O
- Specialty Modules (such as High-Speed Counter, etc)

The time required to write any output image register data to these modules depends on which CPU you are using, the number of modules, and the number of output points.

Remote Module	D2-230	D2-240	D2-250–1	D2-260/D2-262
Overhead	N/A	6.0 µs	1.9 µs	1.9 µs
Per module (with outputs)	N/A	67.5 µs	17.7 µs	17.7 µs
Per output point	N/A	46.0 µs	3.2 µs	3.2 µs

For example, the time required for a D2-240 to write two 8-point output modules (located in a Remote base) would be calculated as follows, where NM is the number of modules and NO is the total number of output points:

Remote I/O

Formula

Time = 6µs + (67.5 µs x NM) + (46µs x NO)

Example

Time = 6µs + (67.5 µs x 2) + (46µs x 16)

Time = 877µs

***NOTE**: This total time is the actual time required for the CPU to update these outputs. This does not include any additional time that is required for the CPU to actually service the particular specialty modules.*

Diagnostics

The DL205 CPUs perform many types of system diagnostics. The amount of time required depends on many things, such as the number of I/O modules installed, etc. The following table shows the minimum and maximum times that can be expected.

Diagnostic Time	D2-230	D2-240	D2-250–1	D2-260/D2-262
Minimum	600.0 µs	422.0 µs	26.8 µs	26.8 µs
Maximum	900.0 µs	855.0 µs	103.0 µs	103.0 µs

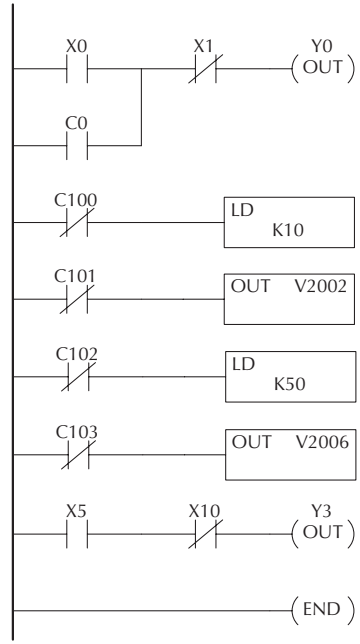
Application Program Execution

The CPU processes the program from the top (address 0) to the END instruction. The CPU executes the program left to right and top to bottom. As each rung is evaluated, the appropriate image register or memory location is updated.

The time required to solve the application program depends on the type and number of instructions used and the amount of execution overhead.

You can add the execution times for all the instructions in your program to find the total program execution time. For example, the execution time for a D2-240 running the program shown would be calculated as follows:

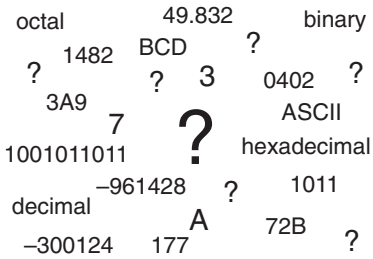
Instruction	Time (µs)
STR X0	1.4
OR C0	1.0
ANDN X1	1.2
OUT Y0	7.95
STRN C100	1.6
LD K10	62.0
STRN C101	1.6
OUT V2002	21.0
STRN C102	1.6
LD K50	62.0
STRN C103	1.6
OUT V2006	21.0
STR X5	1.4
ANDN X10	1.2
OUT Y3	7.95
END	16.0
TOTAL	210.5 µs



Appendix C provides a complete list of instruction execution times for DL205 CPUs.

Program Control Instructions — the D2-240, D2-250–1, D2-260 and D2-262 CPUs offer additional instructions that can change the way the program executes. These instructions include FOR/NEXT loops, Subroutines, and Interrupt Routines. These instructions can interrupt the normal program flow and affect the program execution time. Chapter 5 provides detailed information on how these different types of instructions operate.

PLC Numbering Systems



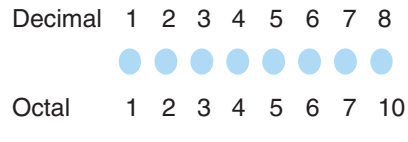
If you are a new PLC user or are using *Direct*LOGIC PLCs for the first time, please take a moment to study how our PLCs use numbers. You'll find that each PLC manufacturer has its own conventions on the use of numbers in their PLCs. Take a moment to familiarize yourself with how numbers are used in *Direct*LOGIC PLCs. The information you learn here applies to all our PLCs.

As any good computer does, PLCs store and manipulate numbers in binary form: ones and zeros. So why do we have to deal with numbers in so many different forms? Numbers have meaning, and some representations are more convenient than others for particular purposes. Sometimes we use numbers to represent a size or amount of something. Other numbers refer to locations or addresses, or to time. In science we attach engineering units to numbers to give a particular meaning (see Appendix H for numbering system details).

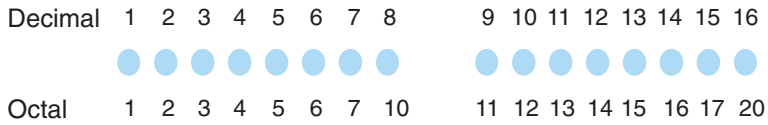
PLC Resources

PLCs offer a fixed number of resources, depending on the model and configuration. We use the word "resources" to include variable memory (V-memory), I/O points, timers, counters, etc. Most modular PLCs allow you to add I/O points in groups of eight. In fact, all the resources of our PLCs are counted in octal. It's easier for computers to count in groups of eight than ten, because eight is an even power of two.

Octal means simply counting in groups of eight. In the figure to the right, there are eight circles. The quantity in decimal is "8," but in octal it is "10" (8 and 9 are not valid in octal). In octal, "10" means 1 group of 8 plus 0 (no individuals).

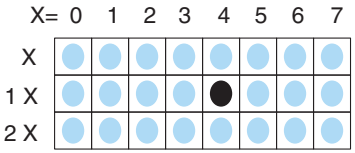


In the figure below, we have two groups of eight circles. Counting in octal we have "20" items, meaning two groups of eight, plus zero individuals Don't say "twenty," say "two–zero octal". This makes a clear distinction between number systems.



After *counting* PLC resources, it's time to access PLC resources (there's a difference). The CPU instruction set accesses resources of the PLC using octal addresses. Octal addresses are the same as octal quantities, except they start counting at zero. The number zero is significant to a computer, so we don't skip it.

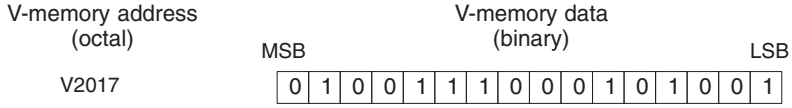
Our circles are in an array of square containers to the right. To access a resource, our PLC instruction will address its location using the octal references shown. If these were counters, "CT14" would access the black circle location.



V–Memory

Variable memory (called "V-memory") stores data for the ladder program and for configuration settings. V-memory locations and V-memory addresses are the same thing, and are numbered in octal. For example, V2073 is a valid location, while V1983 is not valid ("9" and "8" are not valid octal digits).

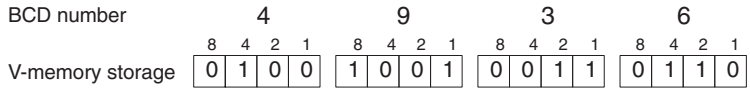
Each V-memory location is one data word wide, meaning 16 bits. For configuration registers, our manuals will show each bit of a V-memory word. The least significant bit (LSB) will be on the right, and the most significant bit (MSB) on the left. We use the word "significant," referring to the relative binary weighting of the bits.



V-memory data is 16-bit binary, but we rarely program the data registers one bit at a time. We use instructions or viewing tools that let us work with binary, decimal, octal, and hexadecimal numbers. All these are converted and stored as binary for us. A frequently-asked question is "How do I tell if a number is binary, octal, BCD, or hex"? The answer is that we usually cannot tell by looking at the data, but it does not really matter. What matters is that the source or mechanism which writes data into a V-memory location and the mechanism which later reads it must both use the same data type (i.e., octal, hex, binary, or whatever). The V-memory location is a storage box, that's all. It does not convert or move the data on its own.

Binary-Coded Decimal Numbers

Since humans naturally count in decimal, we prefer to enter and view PLC data in decimal as well (via operator interfaces). However, computers are more efficient in using pure binary numbers. A compromise solution between the two is Binary-Coded Decimal (BCD) representation. A BCD digit ranges from 0 to 9, and is stored as 4 binary bits (a nibble). This permits each V-memory location to store 4 BCD digits, with a range of decimal numbers from 0000 to 9999.



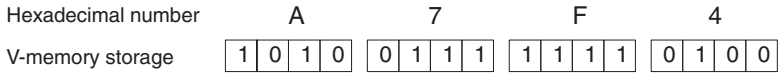
In a pure binary sense, a 16-bit word represents numbers from 0 to 65535. In storing BCD numbers, the range is reduced to 0 to 9999. Many math instructions use BCD data, and *Direct*SOFT and the Handheld Programmer allow us to enter and view data in BCD. Special RLL instructions convert from BCD to binary, or visa–versa.

Hexadecimal Numbers

Hexadecimal numbers are similar to BCD numbers, except they utilize all possible binary values in each 4-bit digit. They are base-16 numbers so we need 16 different digits. To extend our decimal digits 0 through 9, we use A through F as shown.

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

A 4-digit hexadecimal number can represent all 65536 values in a V-memory word. The range is from 0000 to FFFF (hex). PLCs often need this full range for sensor data, etc. Hexadecimal is a convenient way for humans to view full binary data.

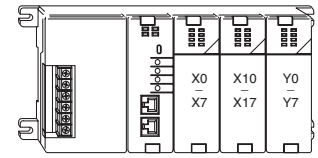


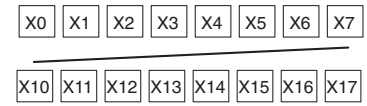
Memory Map

With any PLC system, you generally have many different types of information to process. This includes input device status, output device status, various timing elements, parts counts, etc. It is important to understand how the system represents and stores the various types of data. For example, you need to know how the system identifies input points, output points, data words, etc. The following paragraphs discuss the various memory types used in the DL205 CPUs. A memory map overview for the D2-230, D2-240, D2-250–1, D2-260 and D2-262 CPUs follows the memory descriptions.

Octal Numbering System

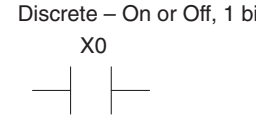
All memory locations or areas are numbered in Octal (base 8). For example, the diagram shows how the octal numbering system works for the discrete input points. Notice the octal system does not contain any numbers with the digits 8 or 9.

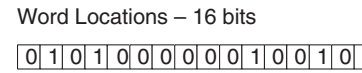




Discrete and Word Locations

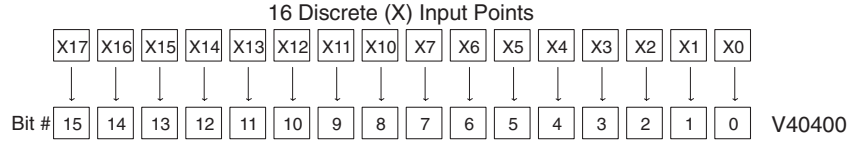
As you examine the different memory types, you'll notice two types of memory in the DL205, discrete and word memory. Discrete memory is one bit that can be either a 1 or a 0. Word memory is referred to as V memory (variable) and is a 16-bit location normally used to manipulate data/numbers, store data/numbers, etc. Some information is automatically stored in V-memory. For example, the timer current values are stored in V-memory.





V–Memory Locations for Discrete Memory Areas

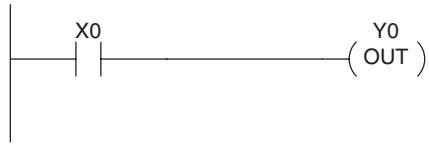
The discrete memory area is for inputs, outputs, control relays, special relays, stages, timer status bits and counter status bits. However, you can also access the bit data types as a V-memory word. Each V-memory location contains 16 consecutive discrete locations. For example, the following diagram shows how the X input points are mapped into V-memory locations.



These discrete memory areas and their corresponding V-memory ranges are listed in the memory table for the D2-230, D2-240, D2-250–1, D2-260 and D2-262 CPUs in this chapter.

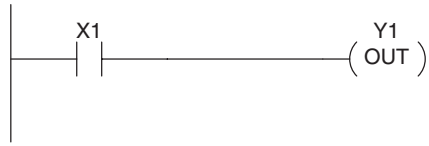
Input Points (X Data Type)

The discrete input points are noted by an X data type. Up to 512 discrete input points are available with the DL205 CPUs. In this example, the output point Y0 will be turned on when input X0 energizes.



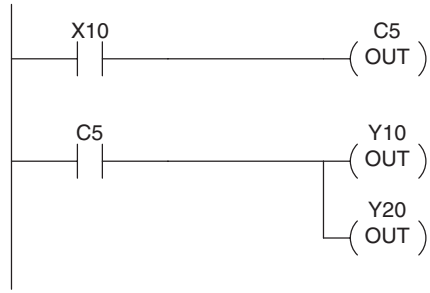
Output Points (Y Data Type)

The discrete output points are noted by a Y data type. Up to 512 discrete output points are available with the DL205 CPUs. In this example, output point Y1 will turn on when input X1 energizes.



Control Relays (C Data Type)

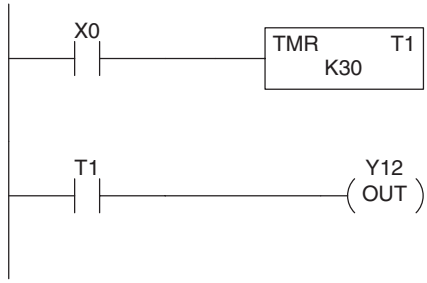
Control relays are discrete bits normally used to control the user program. The control relays do not represent a real world device; that is, they cannot be physically tied to switches, output coils, etc. Control relays are internal to the CPU and can be programmed as discrete inputs or discrete outputs. These locations are used in programming the discrete memory locations (C) or the corresponding word location which has 16 consecutive discrete locations. In this example, memory location C5 will energize when input X10 turns on. The second rung shows a simple example of how to use a control relay as an input.



Timers and Timer Status Bits (T Data Type)

The number of timers available depends on the model of CPU you are using. The tables at the end of this section provide the number of timers for the D2-230, D2-240, D2-250–1, D2-260 and D2-262. Regardless of the number of timers, you have access to timer status bits that reflect the relationship between the current value and the preset value of a specified timer. The timer status bit will be on when the current value is equal to or greater than the preset value of a corresponding timer.

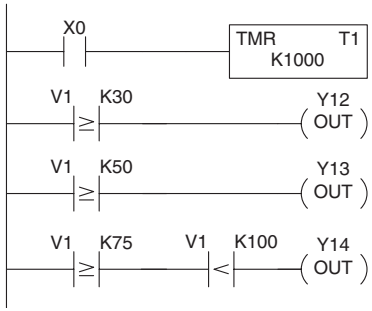
When input X0 turns on, timer T1 will start. When the timer reaches the preset of 3 seconds (K of 30), timer status contact T1 turns on. When T1 turns on, output Y12 turns on.



Timer Current Values (V Data Type)

Some information is automatically stored in V-memory, such as the current values associated with timers. For example, V0 holds the current value for Timer 0, V1 holds the current value for Timer 1, etc. These are 4-digit BCD values.

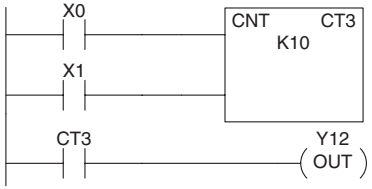
The primary reason for this is programming flexibility. The example shows how you can use relational contacts to monitor several time intervals from a single timer.



Counters and Counter Status Bits (CT Data Type)

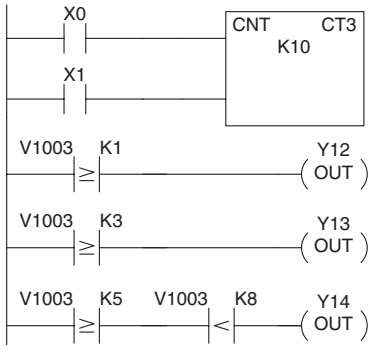
You have access to counter status bits that reflect the relationship between the current value and the preset value of a specified counter. The counter status bit will be on when the current value is equal to or greater than the preset value of a corresponding counter.

Each time contact X0 transitions from off to on, the counter increments by one (If X1 comes on, the counter is reset to zero). When the counter reaches the preset of 10 counts (K of 10), counter status contact CT3 turns on. When CT3 turns on, output Y12 turns on.



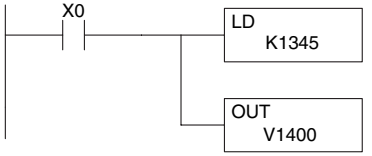
Counter Current Values (V Data Type)

Just like the timers, the counter current values are also automatically stored in V-memory. For example, V1000 holds the current value for Counter CT0, V1001 holds the current value for Counter CT1, etc. These are 4-digit BCD values. The primary reason for this is programming flexibility. The example shows how you can use relational contacts to monitor the counter values.

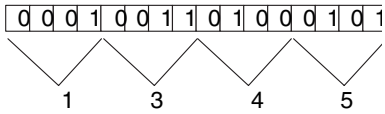


Word Memory (V Data Type)

Word memory is referred to as V-memory (variable) and is a 16-bit location normally used to manipulate data/numbers, store data/numbers, etc. Some information is automatically stored in V-memory. For example, the timer current values are stored in V-memory. The example shows how a four-digit BCD constant is loaded into the accumulator and then stored in a V-memory location.



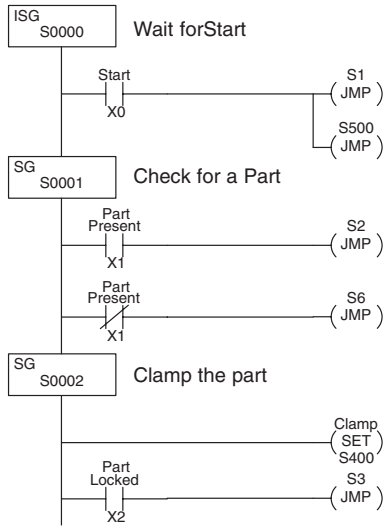
Word Locations – 16 bits



Stages (S Data type)

Stages are used in RLL*PLUS* programs to create a structured program, similar to a flowchart. Each program stage denotes a program segment. When the program segment, or stage, is active, the logic within that segment is executed. If the stage is off, or inactive, the logic is not executed and the CPU skips to the next active stage. (See Chapter 7 for a more detailed description of RLL*PLUS* programming.)

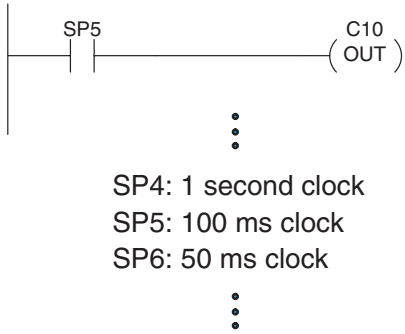
Each stage also has a discrete status bit that can be used as an input to indicate whether the stage is active or inactive. If the stage is active, then the status bit is on. If the stage is inactive, then the status bit is off. This status bit can also be turned on or off by other instructions, such as the SET or RESET instructions. This allows you to easily control stages throughout the program.



Special Relays (SP Data Type)

Special relays are discrete memory locations with pre-defined functionality. There are many different types of special relays. For example, some aid in program development, others provide system operating status information, etc. Appendix D provides a complete listing of the special relays.

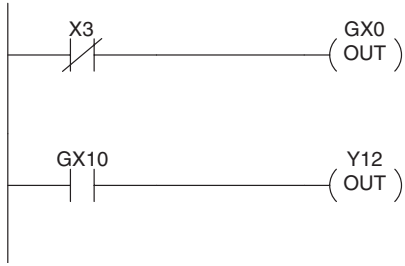
In this example, control relay C10 will energize for 50ms and de–energize for 50 ms because SP5 is a pre–defined relay that will be on for 50ms and off for 50ms.



Remote I/O Points (GX Data Type)

Remote I/O points are represented by global relays. They are generally used only to control remote I/O, but they can be used as normal control relays when remote I/O is not used in the system.

In this example, memory location GX0 represents an output point and memory location GX10 represents an input point.



D2-230 System V-memory

System V-memory	Description of Contents	Default Values/Ranges
V2320–V2377	The default location for multiple preset values for the UP counter.	N/A
V7620–V7627	Locations for DV–1000 operator interface parameters	
V7620	Sets the V-memory location that contains the value.	V0–V2377
V7621	Sets the V-memory location that contains the message.	V0–V2377
V7622	Sets the total number (1 - 16) of V-memory locations to be displayed.	1–16
V7623	Sets the V-memory location that contains the numbers to be displayed.	V0–V2377
V7624	Sets the V-memory location that contains the character code to be displayed.	V0–V2377
V7625	Sets the bit control pointer.	V-memory location for X,Y, or C points used.
V7626	Power Up mode change preset value password.	0,1,2,3,12 Default = 0000
V7627	Reserved for future use.	
V7630	Starting location for the multi–step presets for channel 1. The default value is 2320, which indicates the first value should be obtained from V2320. Since 24 presets are available, the default range is V2320 – V2377. You can change the starting point if necessary.	Default: V2320 Range: V0–V2320
V7631–V7632	Not used	N/A
V7633	Sets the desired mode for the high speed counter, interrupt, pulse catch, pulse train, and input filter (see the D2-CTRINT Manual, D2-CTRIF-M for more information). Location is also used for setting the with/without battery option, enable/disable CPU mode change, and power-up in Run Mode option.	Default: 0000 Lower Byte Range: Range: 0–None 10–Up 40–Interrupt 50–Pulse Catch 60–Filtered discrete In. Upper Byte Range: Bits 8–11, 14,15: Unused Bit 12: With Batt. installed: 0 = disable BATT LED 1 = enable BATT LED Bit 13: Power-up in Run
V7634	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X0 (when D2–CTRINT is installed).	Default: 0000
V7635	Contains set up-information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X1 (when D2–CTRINT is installed).	Default: 0000
V7636	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X2 (when D2–CTRINT is installed).	Default: 0000
V7637	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X3 (when D2–CTRINT is installed).	Default: 0000
V7640–V7642	Additional setup parameters for the DV-1000	
V7640	Timer preset value pointer	V2000–V2377
V7641	Counter preset value pointer	V2000–V2377
V7642	Timer preset block size (high byte) / Counter preset block size (low byte)	1–99

D2-230 System V-memory, Continued

System V-memory	Description of Contents	Default Values/Ranges
V7643–V7647	Not used	N/A
V7751	Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed.	N/A
V7752	I/O Configuration Error — stores the module ID code for the module that does not match the current configuration.	N/A
V7753	I/O Configuration Error — stores the correct module ID code.	
V7754	I/O Configuration Error — identifies the base and slot number.	
V7755	Error code — stores the fatal error code.	N/A
V7756	Error code — stores the major error code.	N/A
V7757	Error code — stores the minor error code.	
V7760–V7764	Module Error — stores the slot number and error code where an I/O error occurs.	
V7765	Scan — stores the total number of scan cycles that have occurred since the last Program Mode to Run Mode transition.	
V7666–V7774	Not used	N/A
V7775	Scan — stores the current scan time (milliseconds).	N/A
V7776	Scan — stores the minimum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds).	N/A
V7777	Scan — stores the maximum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds).	N/A

D2-240 System V-memory

System V-memory	Description of Contents	Default Values/ Ranges
V3630–V3707	The default location for multiple preset values for UP/DWN and UP counter 1 or pulse output function.	N/A
V3710–V3767	The default location for multiple preset values for UP/DWN and UP counter 2.	N/A
V3770–V3773	Not used	N/A
V3774–V3777	Default locations for analog potentiometer data (channels 1–4, respectively).	Range: 0 – 9999
V7620–V7627 V7620 V7621 V7622 V7623 V7624 V7625 V7626 V7627	Locations for DV–1000 operator interface parameters Sets the V-memory location that contains the value. Sets the V-memory location that contains the message. Sets the total number (1 – 16) of V-memory locations to be displayed. Sets the V-memory location that contains the numbers to be displayed. Sets the V-memory location that contains the character code to be displayed. Sets the bit control pointer Power Up Mode Change Preset Value Password.	 V0 – V3760 V0 – V3760 1 – 16 V0 – V3760 V0 – V3760 V-memory location for X, Y, or C points used. 0,1,2,3,12 Default=0000
V7630	Starting location for the multi–step presets for channel 1. Since there are 24 presets available, the default range is V3630 – V3707. You can change the starting point if necessary.	Default: V3630 Range: V0 – V3710
V7631	Starting location for the multi–step presets for channel 2. Since there are 24 presets available, the default range is V3710– V3767. You can change the starting point if necessary.	Default: V3710 Range: V0 – V3710
V7632	Contains the baud rate setting for Port 2. You can use AUX 56 (from the Handheld Programmer) or, use ***Direct***SOFT to set the port parameters if 9600 baud is unacceptable. Also allows you to set a delay time between the assertion of the RTS signal and the transmission of data. This is useful for radio modems that require a key-up delay before data is transmitted. e.g., a value of 0302 sets 10ms Turnaround Delay (TAD) and 9600 baud.	Default: 2 – 9600 baud Lower Byte = Baud Rate Lower Byte Range: 00 = 300 01 = 1200 02 = 9600 03 = 19.2K Upper Byte = Time Delay Upper Byte Range: 01 = 2ms 02 = 5ms 03 = 10ms 04 = 20ms 05 = 50ms 06 = 100ms 07 = 500ms

D2-240 System V-memory, Continued

System V-memory	Description of Contents	Default Values/ Ranges
V7633	Sets the desired mode for the high speed counter, interrupt, pulse catch, pulse train, and input filter (see the D2-CTRINT manual, D2-CTRIF-M, for more information). Location is also used for setting the with/without battery option, enable/disable CPU mode change.	Default: 0000 Lower Byte Range: 0 – None 10 – Up 20 – Up/Dwn. 30 – Pulse Out 40 – Interrupt 50 – Pulse Catch 60 – Filtered Dis. Upper Byte Range: Bits 8 – 11, 15 Unused Bit 12: With Batt. installed: 0 = disable BATT LED 1 = enable BATT LED Bit 13: Power-up in Run Bit 14: Mode chg. enable (K-sequence only)
V7634	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X0 (when D2–CTRINT is installed).	Default: 0000
V7635	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X1 (when D2–CTRINT is installed).	Default: 0000
V7636	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X2 (when D2–CTRINT is installed).	Default: 0000
V7637	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X3 (when D2–CTRINT is installed).	Default: 0000
V7640–V7641	Location for setting the lower and upper limits for the CH1 analog pot.	Default: 0000 Range: 0 – 9999
V7642–V7643	Location for setting the lower and upper limits for the CH2 analog pot.	Default: 0000 Range: 0 – 9999
V7644–V7645	Location for setting the lower and upper limits for the CH3 analog pot.	Default: 0000 Range: 0 – 9999
V7646–V7647	Location for setting the lower and upper limits for the CH4 analog pot.	Default: 0000 Range: 0 – 9999
V7650–V7737	Locations reserved for set-up information used with future options (remote I/O and data communications).	
V7720–V7722	Locations for DV–1000 operator interface parameters.	
V7720	Titled Timer preset value pointer .	V2000–V2377
V7721	Titled Counter preset value pointer.	V2000–V2377
V7722	HiByte-Titled Timer preset block size, LoByte-Titled Counter preset block size.	1–99
V7746	Location contains the battery voltage, accurate to 0.1V. For example, a value of 32 indicates 3.2 volts.	
V7747	Location contains a 10ms counter. This location increments once every 10ms.	
V7751	Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed. If you've used ASCII messages (D2-240 only), then the data label (DLBL) reference number for that message is stored here.	
V7752	I/O configuration Error — stores the module ID code for the module that does not match the current configuration.	

D2-240 System V-memory, Continued

System V-memory	Description of Contents
V7753	I/O Configuration Error — stores the correct module ID code.
V7754	I/O Configuration Error — identifies the base and slot number.
V7755	Error code — stores the fatal error code.
V7756	Error code — stores the major error code.
V7757	Error code — stores the minor error code.
V7760–V7764	Module Error — stores the slot number and error code where an I/O error occurs.
V7765	Scan—stores the number of scan cycles that have occurred since the last Program to Run Mode transition.
V7766	Contains the number of seconds on the clock. (00 to 59).
V7767	Contains the number of minutes on the clock. (00 to 59).
V7770	Contains the number of hours on the clock. (00 to 23).
V7771	Contains the day of the week. (Mon, Tue, etc.).
V7772	Contains the day of the month (1st, 2nd, etc.).
V7773	Contains the month. (01 to 12)
V7774	Contains the year. (00 to 99)
V7775	Scan — stores the current scan time (milliseconds).
V7776	Scan — stores the minimum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds).
V7777	Scan — stores the maximum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds).

D2-250–1 System V-memory (D2-250 also)

System V-memory	Description of Contents	Default Values/ Ranges
V3630–V3707	The default location for multiple preset values for UP/DWN and UP counter 1 or pulse output function	N/A
V3710–V3767	The default location for multiple preset values for UP/DWN and UP counter 2.	N/A
V3770–V3777	Not used	N/A
V7620–V7627 V7620 V7621 V7622 V7623 V7624 V7625 V7626 V7627	Locations for DV–1000 operator interface parameters Sets the V-memory location that contains the value Sets the V-memory location that contains the message Sets the total number (1 – 32) of V-memory locations to be displayed Sets the V-memory location that contains the numbers to be displayed Sets the V-memory location that contains the character code to be displayed Sets the bit control pointer Sets the power up mode Change Preset Value password	 V0 – V3760 V0 – V3760 1 – 32 V0 – V3760 V0 – V3760 V-memory for X, Y, or C 0,1,2,3,12 Default=0000
V7630	Starting location for the multi–step presets for channel 1. Since there are 24 presets available, the default range is V3630 – V3707. You can change the starting point if necessary.	Default: V3630 Range: V0 – V3710
V7631	Starting location for the multi–step presets for channel 2. Since there are 24 presets available, the default range is V3710– V3767. You can change the starting point if necessary.	Default: V3710 Range: V0 – V3710
V7632	Reserved	
V7633	Sets the desired mode for the high-speed counter, interrupt, pulse catch, pulse train, and input filter (see the D2-CTRINT manual, D2-CTRIF-M, for more information). Location is also used for setting the with/without battery option, enable/disable CPU mode change, and power-up in Run Mode option.	Default: 0060 Lower Byte Range: Range: 0 – None 10 – Up 20 – Up/Dwn. 30 – Pulse Out 40 – Interrupt 50 – Pulse Catch 60 – Filtered Dis. Upper Byte Range: Bits 8 – 11, 14–15 Unused Bit 12: With Batt. installed: 0 = disable BATT LED 1 = enable BATT LED Bit 13: Power-up in Run
V7634	Contains set-up information for high-speed counter, interrupt, pulse catch,pulse train output, and input filter for X0 (when D2–CTRINT is installed).	Default: 1006
V7635	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X1 (when D2–CTRINT is installed).	Default: 1006
V7636	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X2 (when D2–CTRINT is installed).	Default: 1006

D2-250–1 System V-memory (D2-250 also), Continued

System V-memory	Description of Contents	Default Values/Ranges
V7637	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X3 (when D2–CTRINT is installed).	Default: 1006
V7640	Loop Table Beginning address.	V1400–V7340 V10000–V17740
V7641	Number of Loops Enabled	1–4
V7642	Error Code – V–memory Error Location for Loop Table.	
V7643–V7647	Reserved.	
V7650	Port 2 End–code setting Setting (A55A), Non–procedure communications start.	
V7651	Port 2 Data format – Non–procedure communications format setting.	
V7652	Port 2 Format Type setting – Non–procedure communications type code setting.	
V7653	Port 2 Terminate–code setting – Non–procedure communications Termination code setting.	
V7654	Port 2 Store V–mem address – Non–procedure communication data store V–Memory address	
V7655	Port 2 Setup area –0–7 Comm protocol (flag 0) 8–15 Comm time out/response delay time (flag 1).	
V7656	Port 2 Setup area – 0–15 Communication (flag 2, flag 3).	
V7657	Port 2: Setup completion code.	
V7660–V7717	Set–up Information – Locations reserved for set-up information used with future options.	
V7720–V7722	Locations for DV–1000 operator interface parameters.	
V7720	Titled Timer preset value pointer.	
V7721	Title Counter preset value pointer.	
V7722	HiByte-Titled Timer preset block size, LoByte-Titled Counter preset block size.	
V7740	Port 2 Communication Auto Reset Timer setup.	
V7741	Output Hold or reset setting: Expansion bases 1 and 2 (D2-250–1).	
V7747	Location contains a 10ms counter. This location increments once every 10ms.	
V7750	Reserved.	
V7751	Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed. If you've used ASCII messages (D2-240 only), then the data label (DLBL) reference number for that message is stored here.	
V7752	I/O configuration Error — stores the module ID code for the module that does not match the current configuration.	
V7753	I/O Configuration Error — stores the correct module ID code.	
V7754	I/O Configuration Error — identifies the base and slot number.	
V7755	Error code — stores the fatal error code.	
V7756	Error code — stores the major error code.	
V7757	Error code — stores the minor error code.	
V7760–V7764	Module Error — stores the slot number and error code where an I/O error occurs.	
V7765	Scan — stores the total number of scan cycles that have occurred since the last Program Mode to Run Mode transition.	

D2-250–1 System V-memory (D2-250 also), Continued

System V-memory	Description of Contents
V7766	Contains the number of seconds on the clock. (00 to 59)
V7767	Contains the number of minutes on the clock. (00 to 59)
V7770	Contains the number of hours on the clock. (00 to 23)
V7771	Contains the day of the week. (Mon, Tue, etc.)
V7772	Contains the day of the month (1st, 2nd, etc.)
V7773	Contains the month. (01 to 12)
V7774	Contains the year. (00 to 99)
V7775	Scan — stores the current scan time (milliseconds)
V7776	Scan — stores the minimum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds)
V7777	Scan — stores the maximum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds)
V36000–36057	Analog pointer method for expansion base 1 (D2-250–1)
V36100–36157	Analog pointer method for expansion base 2 (D2-250–1)
V36400–36427	Analog pointer method for local base
V37700–37737	Port 2: Setup register for Koyo Remote I/O

System CRs	Description of Contents
C740	Completion of setups – ladder logic must turn this relay on when it has finished writing to the Remote I/O setup table.
C741	Erase received data – turning on this flag will erase the received data during a communication error.
C743	Re-start – Turning on this relay will resume after a communications hang-up on an error.
C750 to C757	Setup Error – The corresponding relay will be ON if the setup table contains an error. (C750 = master, C751 = slave 1 C757 = slave 7)
C760 to C767	Communications Ready – The corresponding relay will be ON if the set-up table data is valid. (C760 = master, C761 = slave 1 C767 = slave 7)

D2-260 and D2-262 System V-memory

System V-memory	Description of Contents	Default Values/Ranges
V3630–V3707	The default location for multiple preset values for UP/DWN and UP counter 1 or pulse output function	N/A
V3710–V3767	The default location for multiple preset values for UP/DWN and UP counter 2	N/A
V3770–V3777	Not used	N/A
V7620–V7627 V7620 V7621 V7622 V7623 V7624 V7625 V7626 V7627	Locations for DV–1000 operator interface parameters Sets the V-memory location that contains the value Sets the V-memory location that contains the message Sets the total number (1 – 32) of V-memory locations to be displayed Sets the V-memory location that contains the numbers to be displayed Sets the V-memory location that contains the character code to be displayed Sets the bit control pointer Sets the power up mode Change Preset Value password	 V0 – V3760 V0 – V3760 1 – 32 V0 – V3760 V0 – V3760 V-memory for X, Y, or C 0,1,2,3,12 Default=0000
V7630	Starting location for the multi–step presets for channel 1. Since there are 24 presets available, the default range is V3630 – V3707. You can change the starting point if necessary.	Default: V3630 Range: V0 – V3710
V7631	Starting location for the multi–step presets for channel 2. Since there are 24 presets available, the default range is V3710– V3767. You can change the starting point if necessary.	Default: V3710 Range: V0 – V3710
V7632	Reserved	
V7633	Sets the desired mode for the high-speed counter, interrupt, pulse catch, pulse train, and input filter (see the D2-CTRINT manual, D2-CTRIF-M, for more information). Location is also used for setting the with/without battery option, enable/disable CPU mode change, and power-up in Run Mode option.	Default: 0060 Lower Byte Range: Range: 0 – None 10 – Up 20 – Up/Dwn 30 – Pulse Ou 40 – Interrupt 50 – Pulse Catch 60 – Filtered Dis. Upper Byte Range Bits 8 – 11, 14–15 Unused Bit 12: With Batt. installed: 0 = disable BATT LED 1 = enable BATT LED Bit 13: Power-up in Run
V7634	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X0 (when D2–CTRINT is installed)	Default: 1006
V7635	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X1 (when D2–CTRINT is installed)	Default: 1006
V7636	Contains set-up information for high-speed counter, interrupt, pulse catch, pulse train output, and input filter for X2 (when D2–CTRINT is installed)	Default: 1006

NOTE: *D2-262 supports the Software Interrupt, INT 0, only. Please see the INT instruction in chapter 5 for more information. The D2-262 does not support the D2-CTRINT module.*

D2-260 and D2-262 System V-memory, Continued

System V-memory	Description of Contents	Default Values/ Ranges
V7637	Contains set up information for high speed counter, interrupt, pulse catch, pulse train output, and input filter for X3 (when D2–CTRINT is installed).	Default: 1006
V7640	PID Loop Table Beginning address.	V400–640 V1400–V7340 V10000–V35740
V7641	Number of Loops Enabled.	1–16
V7642	Error Code – V–memory Error Location for Loop Table.	
V7643 - V7647	Reserved.	
V7650	Port 2 End–code Setting (A55A), Non-procedure communications start.	
V7651	Port 2 Data format - Non-procedure communications format setting.	
V7652	Port 2 Format Type setting – Non–procedure communications type code setting.	
V7653	Port 2 Terminate–code setting – Non–procedure communications Termination code setting	
V7654	Port 2 Store V–memory address – Non–procedure communication data store V–Memory address.	
V7655	Port 2 Setup area –0–7 Comm protocol (flag 0) 8–15 Comm time out/response delay time (flag 1)	
V7656	Port 2 Setup area – 0–15 Communication (flag 2, flag 3)	
V7657	Port 2: Setup completion code.	
V7660–V7717	Set–up Information – Locations reserved for set up information used with future options.	
V7720–V7722 V7720 V7721 V7722	Locations for DV-1000 operator interface parameters. Titled Timer preset value pointer. Title Counter preset value pointer. HiByte-Titled Timer preset block size, LoByte-Titled Counter preset block size.	
V7740	Port 2 Communication Auto Reset Timer setup.	
V7741	Output Hold or reset setting: Expansion bases 1 and 2.	
V7742	Output Hold or reset setting: Expansion bases 3 and 4.	
V7747	Location contains a 10ms counter. This location increments once every 10ms.	
V7750	Reserved.	
V7751	Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed. If you've used ASCII messages (D2-240 only), then the data label (DLBL) reference number for that message is stored here.	
V7752	I/O configuration Error — stores the module ID code for the module that does not match the current configuration.	
V7753	I/O Configuration Error — stores the correct module ID code.	
V7754	I/O Configuration Error — identifies the base and slot number.	
V7755	Error code — stores the fatal error code.	
V7756	Error code — stores the major error code.	
V7757	Error code — stores the minor error code.	
V7763–V7764	Module Error — stores the slot number and error code where an I/O error occurs.	
V7765	Scan — stores the total number of scan cycles that have occurred since the last Program Mode to Run Mode transition.	

D2-260 and D2-262 System V-memory, Continued

System V-memory	Description of Contents
V7766	Contains the number of seconds on the clock. (00 to 59).
V7767	Contains the number of minutes on the clock. (00 to 59).
V7770	Contains the number of hours on the clock. (00 to 23).
V7771	Contains the day of the week. (Mon, Tue, etc.).
V7772	Contains the day of the month (1st, 2nd, etc.).
V7773	Contains the month. (01 to 12)
V7774	Contains the year. (00 to 99)
V7775	Scan — stores the current scan time (milliseconds).
V7776	Scan — stores the minimum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds).
V7777	Scan — stores the maximum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds).
V36000–36057	Analog pointer method for expansion base 1
V36100–36157	Analog pointer method for expansion base 2
V36200–36257	Analog pointer method for expansion base 3
V36300–36357	Analog pointer method for expansion base 4
V36400–36427	Analog pointer method for local base
V37700–37737	Port 2: Set-up register for Koyo Remote I/O

The following system control relays (CR) are used for Koyo Remote I/O setup on Communications Port 2.

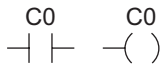
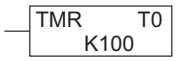
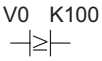
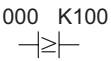
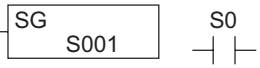
System CR	Description of Contents
C740	Completion of setups – ladder logic must turn this relay on when it has finished writing to the Remote I/O setup table.
C741	Erase received data – turning on this flag will erase the received data during a communication error.
C743	Re-start – Turning on this relay will resume after a communications hang-up on an error.
C750 to C757	Setup Error – The corresponding relay will be ON if the set-up table contains an error. (C750 = master, C751 = slave 1... C757= slave 7
C760 to C767	Communications Ready – The corresponding relay will be ON if the set-up table data is valid. (C760 = master, C761 = slave 1... C767 = slave 7

DL205 Aliases

An alias is an alternate way of referring to certain memory types, such as timer/counter current values, V-memory locations for I/O points, etc., that simplifies understanding the memory address. The use of the alias is optional, but some users may find the alias to be helpful when developing a program. The table below shows how the aliases can be used.

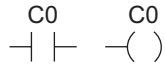
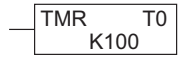
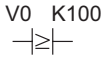
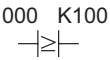
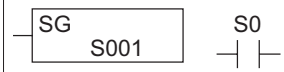
DL205 Aliases		
Address Start	**Alias Start**	**Example**
V0	TA0	V0 is the timer accumulator value for timer 0, therefore, its alias is TA0. TA1 is the alias for V1, etc.
V1000	CTA0	V1000 is the counter accumulator value for counter 0, therefore, its alias is CTA0. CTA1 is the alias for V1001, etc.
V40000	VGX	V40000 is the word memory reference for discrete bits GX0 through GX17, therefore, its alias is VGX0. V40001 is the word memory reference for discrete bits GX20 through GX37, therefore, its alias is VGX20.
V40200	VGY	V40200 is the word memory reference for discrete bits GY0 through GY17, therefore, its alias is VGY0. V40201 is the word memory reference for discrete bits GY20 through GY37, therefore, its alias is VGY20.
V40400	VX0	V40400 is the word memory reference for discrete bits X0 through X17, therefore, its alias is VX0. V40401 is the word memory reference for discrete bits X20 through X37, therefore, its alias is VX20.
V40500	VY0	V40500 is the word memory reference for discrete bits Y0 through Y17, therefore, its alias is VY0. V40501 is the word memory reference for discrete bits Y20 through Y37, therefore, its alias is VY20.
V40600	VC0	V40600 is the word memory reference for discrete bits C0 through C17, therefore, its alias is VC0. V40601 is the word memory reference for discrete bits C20 through C37, therefore, its alias is VC20.
V41000	VS0	V41000 is the word memory reference for discrete bits S0 through S17, therefore, its alias is VS0. V41001 is the word memory reference for discrete bits S20 through S37, therefore, its alias is VS20.
V41100	VT0	V41100 is the word memory reference for discrete bits T0 through T17, therefore, its alias is VT0. V41101 is the word memory reference for discrete bits T20 through T37, therefore, its alias is VT20.
V41140	VCT0	V41140 is the word memory reference for discrete bits CT0 through CT17, therefore, its alias is VCT0. V41141 is the word memory reference for discrete bits CT20 through CT37, therefore, its alias is VCT20.
V41200	VSP0	V41200 is the word memory reference for discrete bits SP0 through SP17, therefore, its alias is VSP0. V41201 is the word memory reference for discrete bits SP20 through SP37, therefore, its alias is VSP20.

D2-230 Memory Map

Memory Type	Discrete Memory Reference (octal)	Word Memory Reference (octal)	Qty. Decimal	Symbol
Input Points	X0 – X177	V40400 – V40407	128[1]	
Output Points	Y0 – Y177	V40500 – V40507	128[1]	
Control Relays	C0 – C377	V40600 – V40617	256	
Special Relays	SP0 – SP117 SP540 – SP577	V41200 – V41204 V41226 – V41227	112	
Timers	T0 – T77		64	
Timer Current Values	None	V0 – V77	64	
Timer Status Bits	T0 – T77	V41100 – V41103	64	
Counters	CT0 – CT77		64	
Counter Current Values	None	V1000 – V1077	64	
Counter Status Bits	CT0 – CT77	V41140 – V41143	64	
Data Words	None	V2000 – V2377	256	None specific, used with many instructions
Data Words Non–volatile	None	V4000 – V4177	128	None specific, used with many instructions
Stages	S0 – S377	V41000 – V41017	256	
System parameters	None	V7620 – V7647 V7750–V7777	48	None specific, used for various purposes

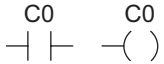
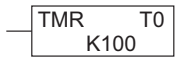
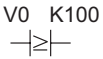
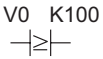
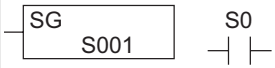
NOTE 1: *The D2-230 systems are limited to 256 discrete I/O points (total) with the present system hardware available. These can be mixed between inputs and output points as necessary.*

D2-240 Memory Map

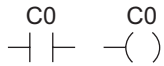
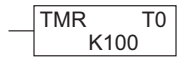
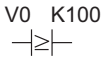
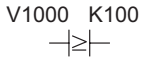
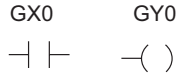
Memory Type	Discrete Memory Reference (octal)	Word Memory Reference(octal)	Qty. Decimal	Symbol
Input Points	X0 – X477	V40400 – V40423	320[1]	
Output Points	Y0 – Y477	V40500 – V40523	320[1]	
Control Relays	C0 – C377	V40600 – V40617	256	
Special Relays	SP0 – SP137 SP540 – SP617	V41200 – V41205 V41226 – V41230	144	
Timers	T0 – T177		128	
Timer Current Values	None	V0 – V177	128	
Timer Status Bits	T0 – T177	V41100 – V41107	128	
Counters	CT0 – CT177		128	
Counter Current Values	None	V1000 – V1177	128	
Counter Status Bits	CT0 – CT177	V41140 – V41147	128	
Data Words	None	V2000 – V3777	1024	None specific, used with many instructions
Data Words Non–volatile	None	V4000 – V4377	256	None specific, used with many instructions
Stages	S0 – S777	V41000 – V41037	512	
System parameters	None	V7620 – V7737 V7746–V7777	106	None specific, used for various purposes

NOTE 1: *The D2-240 systems are limited to 256 discrete I/O points (total) with the present system hardware available. These can be mixed between inputs and output points as necessary.*

D2-250–1 Memory Map (D2-250 also)

Memory Type	Discrete Memory Reference (octal)	Word Memory Reference (octal)	Qty. Decimal	Symbol
Input Points	X0 – X777	V40400 – V40437	512	
Output Points	Y0 – Y777	V40500 – V40537	512	
Control Relays	C0 – C1777	V40600 – V40677	1024	
Special Relays	SP0 – SP777	V41200 – V41237	512	
Timers	T0 – T377		256	
Timer Current Values	None	V0 – V377	256	
Timer Status Bits	T0 – T377	V41100 – V41117	256	
Counters	CT0 – CT177		128	
Counter Current Values	None	V1000 – V1177	128	
Counter Status Bits	CT0 – CT177	V41140 – V41147	128	
Data Words	None	V1400 – V7377 V10000–V17777	7168	None specific, used with many instructions
Stages	S0 – S1777	V41000 – V41077	1024	
System parameters	None	V7400–V7777 V36000–V37777	768	None specific, used for various purposes

D2-260 and D2-262 Memory Map

Memory Type	Discrete Memory Reference (octal)	Word Memory Reference (octal)	Qty. Decimal	Symbol
Input Points	X0 – X1777	V40400 – V40477	1024	
Output Points	Y0 – Y1777	V40500 – V40577	1024	
Control Relays	C0 – C3777	V40600 – V40777	2048	
Special Relays	SP0 – SP777	V41200 – V41237	512	
Timers	T0 – T377		256	
Timer Current Values	None	V0 – V377	256	
Timer Status Bits	T0 – T377	V41100 – V41117	256	
Counters	CT0 – CT377		256	
Counter Current Values	None	V1000 – V1377	256	
Counter Status Bits	CT0 – CT377	V41140 – V41157	256	
Data Words	None	V400 – V777 V1400 – V7377 V10000–V35777	14.6K	None specific, used with many instructions
Stages	S0 – S1777	V41000 – V41077	1024	
Remote Input and Output Points	GX0 – GX3777 GY0 – GY3777	V40000 – V40177 V40200–V40377	2048 2048	
System parameters	None	V7400–V7777 V36000–V37777	1.2K	None specific, used for various purposes

X Input/Y Output Bit Map

This table provides a listing of the individual Input points associated with each V-memory address bit for the D2-230, D2-240, and D2-250–1, D2-260 and D2-262 CPUs. The D2-250–1 ranges apply to the D2-250.

MSB	D2-230/D2-240/D2-250-1/D2-260/D2-262 Input (X) and Output (Y) Points															LSB	X Input Address	Y Output Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000		V40400	V40500
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020		V40401	V40501
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040		V40402	V40502
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060		V40403	V40503
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100		V40404	V40504
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120		V40405	V40505
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140		V40406	V40506
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160		V40407	V40507

MSB	D2-240/D2-250-1/D2-260/D2-262 Input (X) and Output (Y) Points															LSB		
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200		V40410	V40510
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220		V40411	V40511
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240		V40412	V40512
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260		V40413	V40513
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300		V40414	V40514
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320		V40415	V40515
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340		V40416	V40516
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360		V40417	V40517
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400		V40420	V40520
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420		V40421	V40521
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440		V40422	V40522
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460		V40423	V40523

MSB	Additional D2-250-1/D2-260/D2-262 Input (X) and Output (Y) Points															LSB		
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500		V40424	V40524
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520		V40425	V40525
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540		V40426	V40526
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560		V40427	V40527
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600		V40430	V40530
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620		V40431	V40531
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640		V40432	V40532
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660		V40433	V40533
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700		V40434	V40534
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720		V40435	V40535
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740		V40436	V40536
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760		V40437	V40537

X Input/Y Output Bit Map, Continued

MSB	Additional D2-260/D2-262 Input (X) and Output (Y) Points															LSB	X Input Address	Y Output Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
1017	1016	1015	1014	1013	1012	1011	1010	1007	1006	1005	1004	1003	1002	1001	1000	V40440	V40540
1037	1036	1035	1034	1033	1032	1031	1030	1027	1026	1025	1024	1023	1022	1021	1020	V40441	V40541
1057	1056	1055	1054	1053	1052	1051	1050	1047	1046	1045	1044	1043	1042	1041	1040	V40442	V40542
1077	1076	1075	1074	1073	1072	1071	1070	1067	1066	1065	1064	1063	1062	1061	1060	V40443	V40543
1117	1116	1115	1114	1113	1112	1111	1110	1107	1106	1105	1104	1103	1102	1101	1100	V40444	V40544
1137	1136	1135	1134	1133	1132	1131	1130	1127	1126	1125	1124	1123	1122	1121	1120	V40445	V40545
1157	1156	1155	1154	1153	1152	1151	1150	1147	1146	1145	1144	1143	1142	1141	1140	V40446	V40546
1177	1176	1175	1174	1173	1172	1171	1170	1167	1166	1165	1164	1163	1162	1161	1160	V40447	V40547
1217	1216	1215	1214	1213	1212	1211	1210	1207	1206	1205	1204	1203	1202	1201	1200	V40450	V40550
1237	1236	1235	1234	1233	1232	1231	1230	1227	1226	1225	1224	1223	1222	1221	1220	V40451	V40551
1257	1256	1255	1254	1253	1252	1251	1250	1247	1246	1245	1244	1243	1242	1241	1240	V40452	V40552
1277	1276	1275	1274	1273	1272	1271	1270	1267	1266	1265	1264	1263	1262	1261	1260	V40453	V40553
1317	1316	1315	1314	1313	1312	1311	1310	1307	1306	1305	1304	1303	1302	1301	1300	V40454	V40554
1337	1336	1335	1334	1333	1332	1331	1330	1327	1326	1325	1324	1323	1322	1321	1320	V40455	V40555
1357	1356	1355	1354	1353	1352	1351	1350	1347	1346	1345	1344	1343	1342	1341	1340	V40456	V40556
1377	1376	1375	1374	1373	1372	1371	1370	1367	1366	1365	1364	1363	1362	1361	1360	V40457	V40557
1417	1416	1415	1414	1413	1412	1411	1410	1407	1406	1405	1404	1403	1402	1401	1400	V40460	V40560
1437	1436	1435	1434	1433	1432	1431	1430	1427	1426	1425	1424	1423	1422	1421	1420	V40461	V40561
1457	1456	1455	1454	1453	1452	1451	1450	1447	1446	1445	1444	1443	1442	1441	1440	V40462	V40562
1477	1476	1475	1474	1473	1472	1471	1470	1467	1466	1465	1464	1463	1462	1461	1460	V40463	V40563
1517	1516	1515	1514	1513	1512	1511	1510	1507	1506	1505	1504	1503	1502	1501	1500	V40464	V40564
1537	1536	1535	1534	1533	1532	1531	1530	1527	1526	1525	1524	1523	1522	1521	1520	V40465	V40565
1557	1556	1555	1554	1553	1552	1551	1550	1547	1546	1545	1544	1543	1542	1541	1540	V40466	V40566
1577	1576	1575	1574	1573	1572	1571	1570	1567	1566	1565	1564	1563	1562	1561	1560	V40467	V40567
1617	1616	1615	1614	1613	1612	1611	1610	1607	1606	1605	1604	1603	1602	1601	1600	V40470	V40570
1637	1636	1635	1634	1633	1632	1631	1630	1627	1626	1625	1624	1623	1622	1621	1620	V40471	V40571
1657	1656	1655	1654	1653	1652	1651	1650	1647	1646	1645	1644	1643	1642	1641	1640	V40472	V40572
1677	1676	1675	1674	1673	1672	1671	1670	1667	1666	1665	1664	1663	1662	1661	1660	V40473	V40573
1717	1716	1715	1714	1713	1712	1711	1710	1707	1706	1705	1704	1703	1702	1701	1700	V40474	V40574
1737	1736	1735	1734	1733	1732	1731	1730	1727	1726	1725	1724	1723	1722	1721	1720	V40475	V40575
1757	1756	1755	1754	1753	1752	1751	1750	1747	1746	1745	1744	1743	1742	1741	1740	V40476	V40576
1777	1776	1775	1774	1773	1772	1771	1770	1767	1766	1765	1764	1763	1762	1761	1760	V40477	V40577

Control Relay Bit Map

This table provides a listing of the individual control relays associated with each V-memory address bit.

MSB	D2-230/D2-240/D2-250-1/D2-260/D2-262 Control Relays (C)														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	V40600
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V40601
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V40602
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V40603
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V40604
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V40605
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V40606
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V40607
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V40610
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V40611
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V40612
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V40613
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V40614
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V40615
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V40616
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V40617

MSB	Additional D2-250-1/D2-260/D2-262 Control Relays (C)														LSB	Address
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400	V40620
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420	V40621
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440	V40622
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460	V40623
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500	V40624
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520	V40625
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540	V40626
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560	V40627
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600	V40630
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620	V40631
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640	V40632
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660	V40633
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700	V40634
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720	V40635
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740	V40636
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760	V40637

Control Relay Bit Map, Continued

MSB	Additional D2-250-1/D2-260/D2-262 Control Relays (C)														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1017	1016	1015	1014	1013	1012	1011	1010	1007	1006	1005	1004	1003	1002	1001	1000	V40640
1037	1036	1035	1034	1033	1032	1031	1030	1027	1026	1025	1024	1023	1022	1021	1020	V40641
1057	1056	1055	1054	1053	1052	1051	1050	1047	1046	1045	1044	1043	1042	1041	1040	V40642
1077	1076	1075	1074	1073	1072	1071	1070	1067	1066	1065	1064	1063	1062	1061	1060	V40643
1117	1116	1115	1114	1113	1112	1111	1110	1107	1106	1105	1104	1103	1102	1101	1100	V40644
1137	1136	1135	1134	1133	1132	1131	1130	1127	1126	1125	1124	1123	1122	1121	1120	V40645
1157	1156	1155	1154	1153	1152	1151	1150	1147	1146	1145	1144	1143	1142	1141	1140	V40646
1177	1176	1175	1174	1173	1172	1171	1170	1167	1166	1165	1164	1163	1162	1161	1160	V40647
1217	1216	1215	1214	1213	1212	1211	1210	1207	1206	1205	1204	1203	1202	1201	1200	V40650
1237	1236	1235	1234	1233	1232	1231	1230	1227	1226	1225	1224	1223	1222	1221	1220	V40651
1257	1256	1255	1254	1253	1252	1251	1250	1247	1246	1245	1244	1243	1242	1241	1240	V40652
1277	1276	1275	1274	1273	1272	1271	1270	1267	1266	1265	1264	1263	1262	1261	1260	V40653
1317	1316	1315	1314	1313	1312	1311	1310	1307	1306	1305	1304	1303	1302	1301	1300	V40654
1337	1336	1335	1334	1333	1332	1331	1330	1327	1326	1325	1324	1323	1322	1321	1320	V40655
1357	1356	1355	1354	1353	1352	1351	1350	1347	1346	1345	1344	1343	1342	1341	1340	V40656
1377	1376	1375	1374	1373	1372	1371	1370	1367	1366	1365	1364	1363	1362	1361	1360	V40657
1417	1416	1415	1414	1413	1412	1411	1410	1407	1406	1405	1404	1403	1402	1401	1400	V40660
1437	1436	1435	1434	1433	1432	1431	1430	1427	1426	1425	1424	1423	1422	1421	1420	V40661
1457	1456	1455	1454	1453	1452	1451	1450	1447	1446	1445	1444	1443	1442	1441	1440	V40662
1477	1476	1475	1474	1473	1472	1471	1470	1467	1466	1465	1464	1463	1462	1461	1460	V40663
1517	1516	1515	1514	1513	1512	1511	1510	1507	1506	1505	1504	1503	1502	1501	1500	V40664
1537	1536	1535	1534	1533	1532	1531	1530	1527	1526	1525	1524	1523	1522	1521	1520	V40665
1557	1556	1555	1554	1553	1552	1551	1550	1547	1546	1545	1544	1543	1542	1541	1540	V40666
1577	1576	1575	1574	1573	1572	1571	1570	1567	1566	1565	1564	1563	1562	1561	1560	V40667
1617	1616	1615	1614	1613	1612	1611	1610	1607	1606	1605	1604	1603	1602	1601	1600	V40670
1637	1636	1635	1634	1633	1632	1631	1630	1627	1626	1625	1624	1623	1622	1621	1620	V40671
1657	1656	1655	1654	1653	1652	1651	1650	1647	1646	1645	1644	1643	1642	1641	1640	V40672
1677	1676	1675	1674	1673	1672	1671	1670	1667	1666	1665	1664	1663	1662	1661	1660	V40673
1717	1716	1715	1714	1713	1712	1711	1710	1707	1706	1705	1704	1703	1702	1701	1700	V40674
1737	1736	1735	1734	1733	1732	1731	1730	1727	1726	1725	1724	1723	1722	1721	1720	V40675
1757	1756	1755	1754	1753	1752	1751	1750	1747	1746	1745	1744	1743	1742	1741	1740	V40676
1777	1776	1775	1774	1773	1772	1771	1770	1767	1766	1765	1764	1763	1762	1761	1760	V40677

Control Relay Bit Map, Continued

This portion of the table shows additional Control Relays points available with the D2-260 and D2-262.

MSB	Additional D2-260/D2-262 Control Relays (C)															LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
2017	2016	2015	2014	2013	2012	2011	2010	2007	2006	2005	2004	2003	2002	2001	2000	**V40700**
2037	2036	2035	2034	2033	2032	2031	2030	2027	2026	2025	2024	2023	2022	2021	2020	**V40701**
2057	2056	2055	2054	2053	2052	2051	2050	2047	2046	2045	2044	2043	2042	2041	2040	**V40702**
2077	2076	2075	2074	2073	2072	2071	2070	2067	2066	2065	2064	2063	2062	2061	2060	**V40703**
2117	2116	2115	2114	2113	2112	2111	2110	2107	2106	2105	2104	2103	2102	2101	2100	**V40704**
2137	2136	2135	2134	2133	2132	2131	2130	2127	2126	2125	2124	2123	2122	2121	2120	**V40705**
2157	2156	2155	2154	2153	2152	2151	2150	2147	2146	2145	2144	2143	2142	2141	2140	**V40706**
2177	2176	2175	2174	2173	2172	2171	2170	2167	2166	2165	2164	2163	2162	2161	2160	**V40707**
2217	2216	2215	2214	2213	2212	2211	2210	2207	2206	2205	2204	2203	2202	2201	2200	**V40710**
2237	2236	2235	2234	2233	2232	2231	2230	2227	2226	2225	2224	2223	2222	2221	2220	**V40711**
2257	2256	2255	2254	2253	2252	2251	2250	2247	2246	2245	2244	2243	2242	2241	2240	**V40712**
2277	2276	2275	2274	2273	2272	2271	2270	2267	2266	2265	2264	2263	2262	2261	2260	**V40713**
2317	2316	2315	2314	2313	2312	2311	2310	2307	2306	2305	2304	2303	2302	2301	2300	**V40714**
2337	2336	2335	2334	2333	2332	2331	2330	2327	2326	2325	2324	2323	2322	2321	2320	**V40715**
2357	2356	2355	2354	2353	2352	2351	2350	2347	2346	2345	2344	2343	2342	2341	2340	**V40716**
2377	2376	2375	2374	2373	2372	2371	2370	2367	2366	2365	2364	2363	2362	2361	2360	**V40717**
2417	2416	2415	2414	2413	2412	2411	2410	2407	2406	2405	2404	2403	2402	2401	2400	**V40720**
2437	2436	2435	2434	2433	2432	2431	2430	2427	2426	2425	2424	2423	2422	2421	2420	**V40721**
2457	2456	2455	2454	2453	2452	2451	2450	2447	2446	2445	2444	2443	2442	2441	2440	**V40722**
2477	2476	2475	2474	2473	2472	2471	2470	2467	2466	2465	2464	2463	2462	2461	2460	**V40723**
2517	2516	2515	2514	2513	2512	2511	2510	2507	2506	2505	2504	2503	2502	2501	2500	**V40724**
2537	2536	2535	2534	2533	2532	2531	2530	2527	2526	2525	2524	2523	2522	2521	2520	**V40725**
2557	2556	2555	2554	2553	2552	2551	2550	2547	2546	2545	2544	2543	2542	2541	2540	**V40726**
2577	2576	2575	2574	2573	2572	2571	2570	2567	2566	2565	2564	2563	2562	2561	2560	**V40727**
2617	2616	2615	2614	2613	2612	2611	2610	2607	2606	2605	2604	2603	2602	2601	2600	**V40730**
2637	2636	2635	2634	2633	2632	2631	2630	2627	2626	2625	2624	2623	2622	2621	2620	**V40731**
2657	2656	2655	2654	2653	2652	2651	2650	2647	2646	2645	2644	2643	2642	2641	2640	**V40732**
2677	2676	2675	2674	2673	2672	2671	2670	2667	2666	2665	2664	2663	2662	2661	2660	**V40733**
2717	2716	2715	2714	2713	2712	2711	2710	2707	2706	2705	2704	2703	2702	2701	2700	**V40734**
2737	2736	2735	2734	2733	2732	2731	2730	2727	2726	2725	2724	2723	2722	2721	2720	**V40735**
2757	2756	2755	2754	2753	2752	2751	2750	2747	2746	2745	2744	2743	2742	2741	2740	**V40736**
2777	2776	2775	2774	2773	2772	2771	2770	2767	2766	2765	2764	2763	2762	2761	2760	**V40737**

Control Relay Bit Map, Continued

MSB	Additional D2-260/D2-262 Control Relays (C)															LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
3017	3016	3015	3014	3013	3012	3011	3010	3007	3006	3005	3004	3003	3002	3001	3000	**V40740**
3037	3036	3035	3034	3033	3032	3031	3030	3027	3026	3025	3024	3023	3022	3021	3020	**V40741**
3057	3056	3055	3054	3053	3052	3051	3050	3047	3046	3045	3044	3043	3042	3041	3040	**V40742**
3077	3076	3075	3074	3073	3072	3071	3070	3067	3066	3065	3064	3063	3062	3061	3060	**V40743**
3117	3116	3115	3114	3113	3112	3111	3110	3107	3106	3105	3104	3103	3102	3101	3100	**V40744**
3137	3136	3135	3134	3133	3132	3131	3130	3127	3126	3125	3124	3123	3122	3121	3120	**V40745**
3157	3156	3155	3154	3153	3152	3151	3150	3147	3146	3145	3144	3143	3142	3141	3140	**V40746**
3177	3176	3175	3174	3173	3172	3171	3170	3167	3166	3165	3164	3163	3162	3161	3160	**V40747**
3217	3216	3215	3214	3213	3212	3211	3210	3207	3206	3205	3204	3203	3202	3201	3200	**V40750**
3237	3236	3235	3234	3233	3232	3231	3230	3227	3226	3225	3224	3223	3222	3221	3220	**V40751**
3257	3256	3255	3254	3253	3252	3251	3250	3247	3246	3245	3244	3243	3242	3241	3240	**V40752**
3277	3276	3275	3274	3273	3272	3271	3270	3267	3266	3265	3264	3263	3262	3261	3260	**V40753**
3317	3316	3315	3314	3313	3312	3311	3310	3307	3306	3305	3304	3303	3302	3301	3300	**V40754**
3337	3336	3335	3334	3333	3332	3331	3330	3327	3326	3325	3324	3323	3322	3321	3320	**V40755**
3357	3356	3355	3354	3353	3352	3351	3350	3347	3346	3345	3344	3343	3342	3341	3340	**V40756**
3377	3376	3375	3374	3373	3372	3371	3370	3367	3366	3365	3364	3363	3362	3361	3360	**V40757**
3417	3416	3415	3414	3413	3412	3411	3410	3407	3406	3405	3404	3403	3402	3401	3400	**V40760**
3437	3436	3435	3434	3433	3432	3431	3430	3427	3426	3425	3424	3423	3422	3421	3420	**V40761**
3457	3456	3455	3454	3453	3452	3451	3450	3447	3446	3445	3444	3443	3442	3441	3440	**V40762**
3477	3476	3475	3474	3473	3472	3471	3470	3467	3466	3465	3464	3463	3462	3461	3460	**V40763**
3517	3516	3515	3514	3513	3512	3511	3510	3507	3506	3505	3504	3503	3502	3501	3500	**V40764**
3537	3536	3535	3534	3533	3532	3531	3530	3527	3526	3525	3524	3523	3522	3521	3520	**V40765**
3557	3556	3555	3554	3553	3552	3551	3550	3547	3546	3545	3544	3543	3542	3541	3540	**V40766**
3577	3576	3575	3574	3573	3572	3571	3570	3567	3566	3565	3564	3563	3562	3561	3560	**V40767**
3617	3616	3615	3614	3613	3612	3611	3610	3607	3606	3605	3604	3603	3602	3601	3600	**V40770**
3637	3636	3635	3634	3633	3632	3631	3630	3627	3626	3625	3624	3623	3622	3621	3620	**V40771**
3657	3656	3655	3654	3653	3652	3651	3650	3647	3646	3645	3644	3643	3642	3641	3640	**V40772**
3677	3676	3675	3674	3673	3672	3671	3670	3667	3666	3665	3664	3663	3662	3661	3660	**V40773**
3717	3716	3715	3714	3713	3712	3711	3710	3707	3706	3705	3704	3703	3702	3701	3700	**V40774**
3737	3736	3735	3734	3733	3732	3731	3730	3727	3726	3725	3724	3723	3722	3721	3720	**V40775**
3757	3756	3755	3754	3753	3752	3751	3750	3747	3746	3745	3744	3743	3742	3741	3740	**V40776**
3777	3776	3775	3774	3773	3772	3771	3770	3767	3766	3765	3764	3763	3762	3761	3760	**V40777**

Stage Control/Status Bit Map

This table provides a listing of the individual Stage control bits associated with each V-memory address.

MSB	D2-230/D2-240/D2-250-1/D2-260/D2-262 Stage (S) Control Bits														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	V41000
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V41001
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V41002
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V41003
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V41004
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V41005
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V41006
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V41007
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V41010
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V41011
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V41012
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V41013
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V41014
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V41015
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V41016
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V41017

MSB	Additional D2-240/D2-250-1/D2-260/D2-262 Stage (S) Control Bits														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400	V41020
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420	V41021
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440	V41022
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460	V41023
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500	V41024
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520	V41025
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540	V41026
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560	V41027
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600	V41030
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620	V41031
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640	V41032
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660	V41033
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700	V41034
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720	V41035
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740	V41036
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760	V41037

Stage Control/Status Bit Map, Continued

MSB	Additional D2-250-1/D2-260/D2-262 Stage (S) Control Bits															LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1017	1016	1015	1014	1013	1012	1011	1010	1007	1006	1005	1004	1003	1002	1001	1000	**V41040**
1037	1036	1035	1034	1033	1032	1031	1030	1027	1026	1025	1024	1023	1022	1021	1020	**V41041**
1057	1056	1055	1054	1053	1052	1051	1050	1047	1046	1045	1044	1043	1042	1041	1040	**V41042**
1077	1076	1075	1074	1073	1072	1071	1070	1067	1066	1065	1064	1063	1062	1061	1060	**V41043**
1117	1116	1115	1114	1113	1112	1111	1110	1107	1106	1105	1104	1103	1102	1101	1100	**V41044**
1137	1136	1135	1134	1133	1132	1131	1130	1127	1126	1125	1124	1123	1122	1121	1120	**V41045**
1157	1156	1155	1154	1153	1152	1151	1150	1147	1146	1145	1144	1143	1142	1141	1140	**V41046**
1177	1176	1175	1174	1173	1172	1171	1170	1167	1166	1165	1164	1163	1162	1161	1160	**V41047**
1217	1216	1215	1214	1213	1212	1211	1210	1207	1206	1205	1204	1203	1202	1201	1200	**V41050**
1237	1236	1235	1234	1233	1232	1231	1230	1227	1226	1225	1224	1223	1222	1221	1220	**V41051**
1257	1256	1255	1254	1253	1252	1251	1250	1247	1246	1245	1244	1243	1242	1241	1240	**V41052**
1277	1276	1275	1274	1273	1272	1271	1270	1267	1266	1265	1264	1263	1262	1261	1260	**V41053**
1317	1316	1315	1314	1313	1312	1311	1310	1307	1306	1305	1304	1303	1302	1301	1300	**V41054**
1337	1336	1335	1334	1333	1332	1331	1330	1327	1326	1325	1324	1323	1322	1321	1320	**V41055**
1357	1356	1355	1354	1353	1352	1351	1350	1347	1346	1345	1344	1343	1342	1341	1340	**V41056**
1377	1376	1375	1374	1373	1372	1371	1370	1367	1366	1365	1364	1363	1362	1361	1360	**V41057**
1417	1416	1415	1414	1413	1412	1411	1410	1407	1406	1405	1404	1403	1402	1401	1400	**V41060**
1437	1436	1435	1434	1433	1432	1431	1430	1427	1426	1425	1424	1423	1422	1421	1420	**V41061**
1457	1456	1455	1454	1453	1452	1451	1450	1447	1446	1445	1444	1443	1442	1441	1440	**V41062**
1477	1476	1475	1474	1473	1472	1471	1470	1467	1466	1465	1464	1463	1462	1461	1460	**V41063**
1517	1516	1515	1514	1513	1512	1511	1510	1507	1506	1505	1504	1503	1502	1501	1500	**V41064**
1537	1536	1535	1534	1533	1532	1531	1530	1527	1526	1525	1524	1523	1522	1521	1520	**V41065**
1557	1556	1555	1554	1553	1552	1551	1550	1547	1546	1545	1544	1543	1542	1541	1540	**V41066**
1577	1576	1575	1574	1573	1572	1571	1570	1567	1566	1565	1564	1563	1562	1561	1560	**V41067**
1617	1616	1615	1614	1613	1612	1611	1610	1607	1606	1605	1604	1603	1602	1601	1600	**V41070**
1637	1636	1635	1634	1633	1632	1631	1630	1627	1626	1625	1624	1623	1622	1621	1620	**V41071**
1657	1656	1655	1654	1653	1652	1651	1650	1647	1646	1645	1644	1643	1642	1641	1640	**V41072**
1677	1676	1675	1674	1673	1672	1671	1670	1667	1666	1665	1664	1663	1662	1661	1660	**V41073**
1717	1716	1715	1714	1713	1712	1711	1710	1707	1706	1705	1704	1703	1702	1701	1700	**V41074**
1737	1736	1735	1734	1733	1732	1731	1730	1727	1726	1725	1724	1723	1722	1721	1720	**V41075**
1757	1756	1755	1754	1753	1752	1751	1750	1747	1746	1745	1744	1743	1742	1741	1740	**V41076**
1777	1776	1775	1774	1773	1772	1771	1770	1767	1766	1765	1764	1763	1762	1761	1760	**V41077**

Timer and Counter Status Bit Maps

This table provides a listing of the individual timer and counter contacts associated with each V-memory address bit (D2-230, D2-240, D2-250–1, D2-260 and D2-262).

MSB	Timer (T) and Counter (CT) Contacts															LSB	Timer	Counter
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Address
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	V41100	V41140
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V41101	V41141
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V41102	V41142
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V41103	V41143

This portion of the table shows additional Timer and Counter contacts available with the D2-240, D2-250–1, D2-260 and D2-262.

MSB	AdditionalTimer (T) and Counter (CT) Contacts															LSB	Timer	Counter
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Address
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V41104	V41144
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V41105	V41145
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V41106	V41146
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V41107	V41147

This portion of the table shows additional Timer contacts available with the D2-250–1, D2-260 and D2-262.

MSB	Additional Timer (T) Contacts															LSB	Timer
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V41110
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V41111
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V41112
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V41113
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V41114
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V41115
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V41116
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V41117

This portion of the table shows additional Counter contacts available with the D2-260 and D2-262.

MSB	Additional Counter (CT) Contacts															LSB	Counter
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V41150
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V41151
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V41152
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V41153
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V41154
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V41155
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V41156
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V41157

Remote I/O Bit Map

This table provides a listing of the individual remote I/O points associated with each V-memory address bit (D2-260 and D2-262).

MSB	D2-260/D2-262 Remote I/O (GX) and (GY) Points														LSB	GX Address	GY Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	V40000	V40200
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V40001	V40201
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V40002	V40202
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V40003	V40203
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V40004	V40204
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V40005	V40205
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V40006	V40206
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V40007	V40207
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V40010	V40210
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V40011	V40211
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V40012	V40212
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V40013	V40213
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V40004	V40214
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V40015	V40215
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V40016	V40216
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V40007	V40217
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400	V40020	V40220
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420	V40021	V40221
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440	V40022	V40222
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460	V40023	V40223
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500	V40024	V40224
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520	V40025	V40225
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540	V40026	V40226
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560	V40027	V40227
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600	V40030	V40230
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620	V40031	V40231
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640	V40032	V40232
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660	V40033	V40233
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700	V40034	V40234
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720	V40035	V40235
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740	V40036	V40236
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760	V40037	V40237

Remote I/O Bit Map, Continued

MSB	D2-260/D2-262 Remote I/O (GX) and (GY) Points														LSB	GX	GY
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Address
1017	1016	1015	1014	1013	1012	1011	1010	1007	1006	1005	1004	1003	1002	1001	1000	V40040	V40240
1037	1036	1035	1034	1033	1032	1031	1030	1027	1026	1025	1024	1023	1022	1021	1020	V40041	V40241
1057	1056	1055	1054	1053	1052	1051	1050	1047	1046	1045	1044	1043	1042	1041	1040	V40042	V40242
1077	1076	1075	1074	1073	1072	1071	1070	1067	1066	1065	1064	1063	1062	1061	1060	V40043	V40243
1117	1116	1115	1114	1113	1112	1111	1110	1107	1106	1105	1104	1103	1102	1101	1100	V40044	V40244
1137	1136	1135	1134	1133	1132	1131	1130	1127	1126	1125	1124	1123	1122	1121	1120	V40045	V40245
1157	1156	1155	1154	1153	1152	1151	1150	1147	1146	1145	1144	1143	1142	1141	1140	V40046	V40246
1177	1176	1175	1174	1173	1172	1171	1170	1167	1166	1165	1164	1163	1162	1161	1160	V40047	V40247
1217	1216	1215	1214	1213	1212	1211	1210	1207	1206	1205	1204	1203	1202	1201	1200	V40050	V40250
1237	1236	1235	1234	1233	1232	1231	1230	1227	1226	1225	1224	1223	1222	1221	1220	V40051	V40251
1257	1256	1255	1254	1253	1252	1251	1250	1247	1246	1245	1244	1243	1242	1241	1240	V40052	V40252
1277	1276	1275	1274	1273	1272	1271	1270	1267	1266	1265	1264	1263	1262	1261	1260	V40053	V40253
1317	1316	1315	1314	1313	1312	1311	1310	1307	1306	1305	1304	1303	1302	1301	1300	V40054	V40254
1337	1336	1335	1334	1333	1332	1331	1330	1327	1326	1325	1324	1323	1322	1321	1320	V40055	V40255
1357	1356	1355	1354	1353	1352	1351	1350	1347	1346	1345	1344	1343	1342	1341	1340	V40056	V40256
1377	1376	1375	1374	1373	1372	1371	1370	1367	1366	1365	1364	1363	1362	1361	1360	V40057	V40257
1417	1416	1415	1414	1413	1412	1411	1410	1407	1406	1405	1404	1403	1402	1401	1400	V40060	V40260
1437	1436	1435	1434	1433	1432	1431	1430	1427	1426	1425	1424	1423	1422	1421	1420	V40061	V40261
1457	1456	1455	1454	1453	1452	1451	1450	1447	1446	1445	1444	1443	1442	1441	1440	V40062	V40262
1477	1476	1475	1474	1473	1472	1471	1470	1467	1466	1465	1464	1463	1462	1461	1460	V40063	V40263
1517	1516	1515	1514	1513	1512	1511	1510	1507	1506	1505	1504	1503	1502	1501	1500	V40064	V40264
1537	1536	1535	1534	1533	1532	1531	1530	1527	1526	1525	1524	1523	1522	1521	1520	V40065	V40265
1557	1556	1555	1554	1553	1552	1551	1550	1547	1546	1545	1544	1543	1542	1541	1540	V40066	V40266
1577	1576	1575	1574	1573	1572	1571	1570	1567	1566	1565	1564	1563	1562	1561	1560	V40067	V40267
1617	1616	1615	1614	1613	1612	1611	1610	1607	1606	1605	1604	1603	1602	1601	1600	V40070	V40270
1637	1636	1635	1634	1633	1632	1631	1630	1627	1626	1625	1624	1623	1622	1621	1620	V40071	V40271
1657	1656	1655	1654	1653	1652	1651	1650	1647	1646	1645	1644	1643	1642	1641	1640	V40072	V40272
1677	1676	1675	1674	1673	1672	1671	1670	1667	1666	1665	1664	1663	1662	1661	1660	V40073	V40273
1717	1716	1715	1714	1713	1712	1711	1710	1707	1706	1705	1704	1703	1702	1701	1700	V40074	V40274
1737	1736	1735	1734	1733	1732	1731	1730	1727	1726	1725	1724	1723	1722	1721	1720	V40075	V40275
1757	1756	1755	1754	1753	1752	1751	1750	1747	1746	1745	1744	1743	1742	1741	1740	V40076	V40276
1777	1776	1775	1774	1773	1772	1771	1770	1767	1766	1765	1764	1763	1762	1761	1760	V40077	V40277

Remote I/O Bit Map, Continued

MSB	D2-260/D2-262 Remote I/O (GX) and (GY) Points															LSB	GX	GY
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Address
2017	2016	2015	2014	2013	2012	2011	2010	2007	2006	2005	2004	2003	2002	2001	2000	V40100	V40300
2037	2036	2035	2034	2033	2032	2031	2030	2027	2026	2025	2024	2023	2022	2021	2020	V40101	V40301
2057	2056	2055	2054	2053	2052	2051	2050	2047	2046	2045	2044	2043	2042	2041	2040	V40102	V40302
2077	2076	2075	2074	2073	2072	2071	2070	2067	2066	2065	2064	2063	2062	2061	2060	V40103	V40303
2117	2116	2115	2114	2113	2112	2111	2110	2107	2106	2105	2104	2103	2102	2101	2100	V40104	V40304
2137	2136	2135	2134	2133	2132	2131	2130	2127	2126	2125	2124	2123	2122	2121	2120	V40105	V40305
2157	2156	2155	2154	2153	2152	2151	2150	2147	2146	2145	2144	2143	2142	2141	2140	V40106	V40306
2177	2176	2175	2174	2173	2172	2171	2170	2167	2166	2165	2164	2163	2162	2161	2160	V40107	V40307
2217	2216	2215	2214	2213	2212	2211	2210	2207	2206	2205	2204	2203	2202	2201	2200	V40110	V40310
2237	2236	2235	2234	2233	2232	2231	2230	2227	2226	2225	2224	2223	2222	2221	2220	V40111	V40311
2257	2256	2255	2254	2253	2252	2251	2250	2247	2246	2245	2244	2243	2242	2241	2240	V40112	V40312
2277	2276	2275	2274	2273	2272	2271	2270	2267	2266	2265	2264	2263	2262	2261	2260	V40113	V40313
2317	2316	2315	2314	2313	2312	2311	2310	2307	2306	2305	2304	2303	2302	2301	2300	V40114	V40314
2337	2336	2335	2334	2333	2332	2331	2330	2327	2326	2325	2324	2323	2322	2321	2320	V40115	V40315
2357	2356	2355	2354	2353	2352	2351	2350	2347	2346	2345	2344	2343	2342	2341	2340	V40116	V40316
2377	2376	2375	2374	2373	2372	2371	2370	2367	2366	2365	2364	2363	2362	2361	2360	V40117	V40317
2417	2416	2415	2414	2413	2412	2411	2410	2407	2406	2405	2404	2403	2402	2401	2400	V40120	V40320
2437	2436	2435	2434	2433	2432	2431	2430	2427	2426	2425	2424	2423	2422	2421	2420	V40121	V40321
2457	2456	2455	2454	2453	2452	2451	2450	2447	2446	2445	2444	2443	2442	2441	2440	V40122	V40322
2477	2476	2475	2474	2473	2472	2471	2470	2467	2466	2465	2464	2463	2462	2461	2460	V40123	V40323
2517	2516	2515	2514	2513	2512	2511	2510	2507	2506	2505	2504	2503	2502	2501	2500	V40124	V40324
2537	2536	2535	2534	2533	2532	2531	2530	2527	2526	2525	2524	2523	2522	2521	2520	V40125	V40325
2557	2556	2555	2554	2553	2552	2551	2550	2547	2546	2545	2544	2543	2542	2541	2540	V40126	V40326
2577	2576	2575	2574	2573	2572	2571	2570	2567	2566	2565	2564	2563	2562	2561	2560	V40127	V40327
2617	2616	2615	2614	2613	2612	2611	2610	2607	2606	2605	2604	2603	2602	2601	2600	V40130	V40330
2637	2636	2635	2634	2633	2632	2631	2630	2627	2626	2625	2624	2623	2622	2621	2620	V40131	V40331
2657	2656	2655	2654	2653	2652	2651	2650	2647	2646	2645	2644	2643	2642	2641	2640	V40132	V40332
2677	2676	2675	2674	2673	2672	2671	2670	2667	2666	2665	2664	2663	2662	2661	2660	V40133	V40333
2717	2716	2715	2714	2713	2712	2711	2710	2707	2706	2705	2704	2703	2702	2701	2700	V40134	V40334
2737	2736	2735	2734	2733	2732	2731	2730	2727	2726	2725	2724	2723	2722	2721	2720	V40135	V40335
2757	2756	2755	2754	2753	2752	2751	2750	2747	2736	2735	2734	2733	2732	2731	2730	V40136	V40336
2777	2776	2775	2774	2773	2772	2771	2770	2767	2766	2765	2764	2763	2762	2761	2760	V40137	V40337

Remote I/O Bit Map, Continued

MSB	D2-260/D2-262 Remote I/O (GX) and (GY) Points															LSB	GX	GY
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Address
3017	3016	3015	3014	3013	3012	3011	3010	3007	3006	3005	3004	3003	3002	3001	3000	V40140	V40340
3037	3036	3035	3034	3033	3032	3031	3030	3027	3026	3025	3024	3023	3022	3021	3020	V40141	V40341
3057	3056	3055	3054	3053	3052	3051	3050	3047	3046	3045	3044	3043	3042	3041	3040	V40142	V40342
3077	3076	3075	3074	3073	3072	3071	3070	3067	3066	3065	3064	3063	3062	3061	3060	V40143	V40343
3117	3116	3115	3114	3113	3112	3111	3110	3107	3106	3105	3104	3103	3102	3101	3100	V40144	V40344
3137	3136	3135	3134	3133	3132	3131	3130	3127	3126	3125	3124	3123	3122	3121	3120	V40145	V40345
3157	3156	3155	3154	3153	3152	3151	3150	3147	3146	3145	3144	3143	3142	3141	3140	V40146	V40346
3177	3176	3175	3174	3173	3172	3171	3170	3167	3166	3165	3164	3163	3162	3161	3160	V40147	V40347
3217	3216	3215	3214	3213	3212	3211	3210	3207	3206	3205	3204	3203	3202	3201	3200	V40150	V40350
3237	3236	3235	3234	3233	3232	3231	3230	3227	3226	3225	3224	3223	3222	3221	3220	V40151	V40351
3257	3256	3255	3254	3253	3252	3251	3250	3247	3246	3245	3244	3243	3242	3241	3240	V40152	V40352
3277	3276	3275	3274	3273	3272	3271	3270	3267	3266	3265	3264	3263	3262	3261	3260	V40153	V40353
3317	3316	3315	3314	3313	3312	3311	3310	3307	3306	3305	3304	3303	3302	3301	3300	V40154	V40354
3337	3336	3335	3334	3333	3332	3331	3330	3327	3326	3325	3324	3323	3322	3321	3320	V40155	V40355
3357	3356	3355	3354	3353	3352	3351	3350	3347	3346	3345	3344	3343	3342	3341	3340	V40156	V40356
3377	3376	3375	3374	3373	3372	3371	3370	3367	3366	3365	3364	3363	3362	3361	3360	V40157	V40357
3417	3416	3415	3414	3413	3412	3411	3410	3407	3406	3405	3404	3403	3402	3401	3400	V40160	V40360
3437	3436	3435	3434	3433	3432	3431	3430	3427	3426	3425	3424	3423	3422	3421	3420	V40161	V40361
3457	3456	3455	3454	3453	3452	3451	3450	3447	3446	3445	3444	3443	3442	3441	3440	V40162	V40362
3477	3476	3475	3474	3473	3472	3471	3470	3467	3466	3465	3464	3463	3462	3461	3460	V40163	V40363
3517	3516	3515	3514	3513	3512	3511	3510	3507	3506	3505	3504	3503	3502	3501	3500	V40164	V40364
3537	3536	3535	3534	3533	3532	3531	3530	3527	3526	3525	3524	3523	3522	3521	3520	V40165	V40365
3557	3556	3555	3554	3553	3552	3551	3550	3547	3546	3545	3544	3543	3542	3541	3540	V40166	V40366
3577	3576	3575	3574	3573	3572	3571	3570	3567	3566	3565	3564	3563	3562	3561	3560	V40167	V40367
3617	3616	3615	3614	3613	3612	3611	3610	3607	3606	3605	3604	3603	3602	3601	3600	V40170	V40370
3637	3636	3635	3634	3633	3632	3631	3630	3627	3626	3625	3624	3623	3622	3621	3620	V40171	V40371
3657	3656	3655	3654	3653	3652	3651	3650	3647	3646	3645	3644	3643	3642	3641	3640	V40172	V40372
3677	3676	3675	3674	3673	3672	3671	3670	3667	3666	3665	3664	3663	3662	3661	3660	V40173	V40373
3717	3716	3715	3714	3713	3712	3711	3710	3707	3706	3705	3704	3703	3702	3701	3700	V40174	V40374
3737	3736	3735	3734	3733	3732	3731	3730	3727	3726	3725	3724	3723	3722	3721	3720	V40175	V40375
3757	3756	3755	3754	3753	3752	3751	3750	3747	3746	3745	3744	3743	3742	3741	3740	V40176	V40376
3777	3776	3775	3774	3773	3772	3771	3770	3767	3766	3765	3764	3763	3762	3761	3760	V40177	V40377

System Design and Configuration

In This Chapter...

DL205 System Design Strategies

I/O System Configurations

The DL205 PLCs offer the following ways to add I/O to the system:

- **Local I/O** – consists of I/O modules located in the same base as the CPU.
- **Local Expansion I/O** – consists of I/O modules in expansion bases located close to the CPU local base. Expansion cables connect the expansion bases and CPU base in daisy–chain format.
- **Ethernet Remote Master** – provides a low-cost, high-speed Ethernet Remote I/O link to Ethernet Remote Slave I/O.
- **Ethernet Base Controller** – provides a low-cost, high-speed Ethernet link between a network master to AutomationDirect Ethernet Remote Slave I/O.
- **Remote I/O** – consists of I/O modules located in bases which are serially connected to the local CPU base through a Remote Master module, or may connect directly to the bottom port on a D2-250-1, D2-260 or D2-262 CPU.

A DL205 system can be developed using many different arrangements of these configurations. All I/O configurations use the standard complement of DL205 I/O modules and bases. Local expansion requires using (–1) bases.

Networking Configurations

The DL205 PLCs offers the following way to add networking to the system:

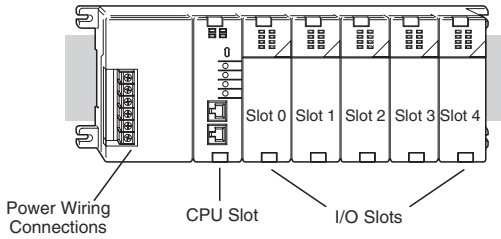
- **Ethernet Communications Module** – connects DL205 systems (D2-240, D2-250-1, D2-260 or D2-262 CPUs only) and DL405 CPU systems in high–speed, peer–to–peer networks. Any PLC can initiate communications with any other PLC when using either the ECOM or ECOM100 modules.
- **Data Communications Module** – connects a DL205 (D2-240, D2-250–1, D2-260 or D2-262 only) system to devices using the DirectNET protocol, or connects as a slave to a Modbus RTU network.
- **D2-250-1 Communications Port** – The D2-250-1 CPU has a 15–Pin connector on Port 2 that provides a built–in Modbus RTU or DirectNET master/slave connection.
- **D2-260/D2-262 Communications Port** – The D2-260 and D2-262 CPUs have a 15–Pin connector on Port 2 that provides a built–in DirectNET master/slave or Modbus RTU master/slave connection with more Modbus function codes than the D2-250-1. The D2-260 and D2-262 MRX and MWX instructions allow you to enter native Modbus addressing in your ladder program with no need to perform octal to decimal conversions. Port 2 can also be used for ASCII IN or ASCII OUT communications.

Module/Unit	Master	Slave
D2-240 CPU		*Direct*Net, K–Sequence
D2-250-1 CPU	*Direct*Net, Modbus RTU	*Direct*Net, K–Sequence, Modbus RTU
D2-260, D2-262 CPU	*Direct*Net, Modbus RTU, ASCII	*Direct*Net, K–Sequence, Modbus RTU, ASCII
ECOM	Ethernet	Ethernet
ECOM100	Ethernet, Modbus TCP	Ethernet, Modbus TCP
DCM	*Direct*Net	*Direct*Net, K–Sequence, Modbus RTU

Module Placement

Slot Numbering

The DL205 bases each provide different numbers of slots for use with the I/O modules. You may notice the bases refer to 3-slot, 4-slot, etc. One of the slots is dedicated to the CPU, so you always have one less I/O slot. For example, you have five I/O slots with a 6-slot base. The I/O slots are numbered 0 – 4. The CPU slot always contains a CPU or a base controller (EBC) or Remote Slave and is not numbered.



Module Placement Restrictions

The following table lists the valid locations for all types of modules in a DL205 system.

Module/Unit	Local CPU Base	Local Expansion Base	Remote I/O Base
CPUs	CPU Slot Only		
DC Input Modules	√	√	√
AC Input Modules	√	√	√
DC Output Modules	√	√	√
AC Output Modules	√	√	√
Relay Output Modules	√	√	√
Analog Input and Output Modules	√	√	√
Local Expansion			
Base Expansion Unit	√	√	
Base Controller Module		CPU Slot Only	
Serial Remote I/O			
Remote Master	√ (not Slot 0)		
Remote Slave Unit			CPU Slot Only
Ethernet Remote Master	√ (not Slot 0)		
Ethernet Slave (EBC)	CPU Slot Only		
CPU Interface			
Ethernet Base Controller	CPU Slot Only		CPU Slot Only*
WinPLC	CPU Slot Only		
DeviceNet	CPU Slot Only		√
Profibus	CPU Slot Only		
SDS	CPU Slot Only		
Specialty Modules			
Counter Interface (CTRINT)	Slot 0 Only		
Counter I/O (CTRIO)	√		√ *
Data Communications	√ (not Slot 0)		
Ethernet Communications	√ (not Slot 0)		
BASIC CoProcessor	√ (not Slot 0)		
Simulator	√	√	√
Filler	√	√	√

*When used in H2–ERM(100) Ethernet Remote I/O systems.

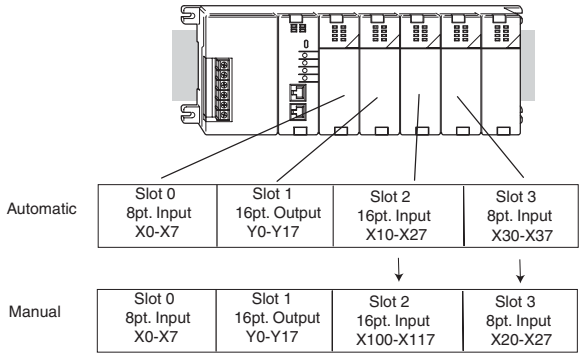
Automatic I/O Configuration

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The DL205 CPUs automatically detect any installed I/O modules (including specialty modules) at powerup, and establish the correct I/O configuration and addresses. This applies to modules located in local and local expansion I/O bases. For most applications, you will never have to change the configuration.

I/O addresses use octal numbering, starting at X0 and Y0 in the slot next to the CPU. The addresses are assigned in groups of 8 or 16, depending on the number of points for the I/O module. The discrete input and output modules can be mixed in any order, but there may be restrictions placed on some specialty modules. The following diagram shows the I/O numbering convention for an example system.

Both the Handheld Programmer and *Direct*SOFT provide AUX functions that allow you to automatically configure the I/O. For example, with the Handheld Programmer AUX 46 executes an automatic configuration, which allows the CPU to examine the installed modules and determine the I/O configuration and addressing. With *Direct*SOFT, the PLC Configure I/O menu option would be used.



Manual I/O Configuration

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

It may never become necessary, but D2-250–1, D2-260 and D2-262 CPUs allow manual I/O address assignments for any I/O slot(s) in local or local expansion bases. You can manually modify an auto configuration to match arbitrary I/O numbering. For example, two adjacent input modules can have starting addresses at X20 and X200. Use *Direct*SOFT PLC Configure I/O menu option to assign manual I/O address.

In automatic configuration, the addresses are assigned on 8-point boundaries. Manual configuration, however, assumes that all modules are at least 16 points, so you can only assign addresses that are a multiple of 20 (octal). For example, X30 and Y50 are not valid starting addresses. You can still use 8-point modules, but 16 addresses will be assigned and the upper eight addresses will be unused.

WARNING: If you manually configure an I/O slot, the I/O addressing for the other modules may change. This is because the D2-250–1, D2-260 and D2-262 CPUs do not allow you to assign duplicate I/O addresses. You must always correct any I/O configuration errors before you place the CPU in RUN mode. Uncorrected errors can cause unpredictable machine operation that can result in a risk of personal injury or damage to equipment.

Removing a Manual Configuration

After a manual configuration, the system will automatically retain the new I/O addresses through a power cycle. You can remove (overwrite) any manual configuration changes by changing all of the manually configured addresses back to automatic.

Power–On I/O Configuration Check

The DL205 CPUs can also be set to automatically check the I/O configuration on power-up. By selecting this feature, you can detect any changes that may have occurred while the power was disconnected. For example, if someone places an output module in a slot that previously held an input module, the CPU will not go into RUN mode and the configuration check will detect the change and print a message on the Handheld Programmer or *Direct*SOFT screen (use AUX 44 on the HPP to enable the configuration check).

If the system detects a change in the PLC/Setup/I/O configuration check at power-up, error code E252 will be generated. You can use AUX 42 (HPP) or *Direct*SOFT I/O diagnostics to determine the exact base and slot location where the change occurred. When a configuration error is generated, you may actually want to use the new I/O configuration. For example, you may have intentionally changed an I/O module to use with a program change. You can use PLC/Diagnostics/I/O Diagnostics in *Direct*Soft or AUX 45 to select the new configuration, or, keep the existing configuration stored in memory.

WARNING: You should always correct any I/O configuration errors before you place the CPU into RUN mode. Uncorrected errors can cause unpredictable machine operation that can result in a risk of personal injury or damage to equipment.

WARNING: Verify that the I/O configuration being selected will work properly with the CPU program. Always correct any I/O configuration errors before placing the CPU in RUN mode. Uncorrected errors can cause unpredictable machine operation that can result in a risk of personal injury or damage to equipment.

I/O Points Required for Each Module

Each type of module requires a certain number of I/O points. This is also true for some specialty modules, such as analog, counter interface, etc.

DC Input Modules	Number of I/O Pts. Required
D2–08ND3	8 Input
D2–16ND3–2	16 Input
D2–32ND3(–2)	32 Input
AC Input Modules	
D2–08NA–1	8 Input
D2–08NA–2	8 Input
D2–16NA	16 Input
DC Output Modules	
D2–04TD1	8 Output (Only the first four points are used)
D2–08TD1	8 Output
D2–16TD1–2 (2-2)	16 Output
D2–16TD1(2)P	16 Output
D2–32TD1(–2)	32 Output
AC Output Modules	
D2–08TA	8 Output
F2–08TA	8 Output
D2–12TA	16 Output (See note 1)
Relay Output Modules	
D2–04TRS	8 Output (Only the first four points are used)
D2–08TR	8 Output
F2–08TRS	8 Output
F2–08TR	8 Output
D2–12TR	16 Output (See note 1)
Combination Modules	
D2–08CDR	8 In, 8 Out (Only the first four points are used for each type)
Analog Modules	
F2–04AD–1 & 1L	16 Input
F2–04AD–2 & 2L	16 Input
F2–08AD–1	16 Input
F2–02DA–1 & 1L	16 Output
F2–02DA–2 & 2L	16 Output
F2–08DA–1	16 Output
F2–08DA–2	16 Output
F2–02DAS–1	32 Output
F2–02DAS–2	32 Output
F2–4AD2DA	16 Input & 16 Output
F2–8AD4DA-1	32 Input & 32 Output
F2–8AD4DA-2	32 Input & 32 Output
F2–04RTD	32 Input
F2–04THM	32 Input

Specialty Modules, etc.	Number of I/O Pts. Required
H2–ECOM(–F)	None
D2–DCM	None
H2–ERM(100,–F)	None
H2–EBC(–F)	None
D2–RMSM	None
D2–RSSS	None
F2–CP128	None
H2–CTRIO(2)	None
D2–CTRINT*	8 Input 8 Output
F2–DEVNETS–1	None
H2–PBC	None
F2–SDS–1	None
D2–08SIM	8 Input
D2-EM	None
D2-CM	None
H2-ECOM(100)	None

*D2-CTRINT not supported by D2-262.

***NOTE**: 12pt modules consume 16 points. The first 6 points are assigned, two are skipped, and then the next 6 points are assigned. For example, a D2–12TA installed in slot 0 would use Y0–Y5, and Y-10-Y15. Y6–Y7 and Y16–Y17 would be unused.*

Calculating the Power Budget

Managing Your Power Resource

When you determine the types and quantities of I/O modules you will be using in the DL205 system, it is important to remember there is a limited amount of power available from the power supply. We have provided a chart to help you easily see the amount of power available with each base. The following chart will help you calculate the amount of power you need with your I/O selections. At the end of this section is an example of power budgeting and a worksheet for your own calculations.

If the I/O you choose exceeds the maximum power available from the power supply, you may need to use local expansion bases or remote I/O bases.

WARNING: It is extremely important to calculate the power budget. If you exceed the power budget, the system may operate in an unpredictable manner, which may result in a risk of personal injury or equipment damage.

CPU Power Specifications

The following chart shows the amount of current available for the two voltages supplied from the DL205 base. Use these currents when calculating the power budget for your system. The Auxiliary 24V Power Source mentioned in the table is a connection at the base terminal strip allowing you to connect to devices or DL205 modules that require 24VDC.

Bases	5V Current Supplied	Auxiliary 24VDC Current Supplied
D2–03B–1	2600mA	300mA
D2–04B–1		
D2–06B–1		
D2–09B–1		
D2–03BDC1–1	2600mA	None
D2–04BDC1–1		
D2–06BDC1–1		
D2–09BDC1–1		
D2–06BDC2–1	2600mA	300mA
D2–09BDC2–1		

Module Power Requirements

Use the power requirements shown on the next page to calculate the power budget for your system. If an External 24VDC power supply is required, the external 24VDC from the base power supply may be used as long as the power budget is not exceeded.

Power Consumed			Power Consumed		
Device	**5V (mA)**	**24V Auxilliary (mA)**	**Device**	**5V (mA)**	**24V Auxilliary (mA)**
CPUs			**Combination Modules**		
D2–230	120	0	D2–08CDR	200	0
D2–240	120	0	**Specialty Modules**		
D2–250–1	330	0	H2–PBC	530	0
D2–260	330	0	H2–ECOM	450	0
D2–262	336	0	H2–ECOM100	300	0
DC Input Modules			H2–ECOM-F	640	0
D2–08ND3	50	0	H2–ERM(100)	320	0
D2–16ND3–2	100	0	H2–ERM–F	450	0
D2–32ND3(–2)	25	0	H2–EBC	320	0
AC Input Modules			H2–EBC–F	450	0
D2–08NA–1	50	0	H2–CTRIO(2)	275	0
D2–08NA–2	100	0	D2–DCM	300	0
D2–16NA	100	0	D2–RMSM	200	0
DC Output Modules			D2–RSSS	150	0
D2–04TD1	60	20	D2–CTRINT	50*	0
D2–08TD1(–2)	100	0	D2–08SIM	50	0
D2–16TD1–2	200	80	D2–CM	100	0
D2–16TD2–2	200	0	D2–EM	130	0
D2–32TD1(–2)	350	0	F2–CP128	235	0
AC Output Modules			F2–DEVNETS–1	160	0
D2–08TA	250	0	F2–SDS–1	160	0
F2–08TA	250	0			
D2–12TA	350	0			
Relay Output Modules					
D2–04TRS	250	0			
D2–08TR	250	0			
F2–08TRS	670	0			
F2–08TR	670	0			
D2–12TR	450	0			
Analog Modules					
F2–04AD–1	50	80	F2–02DAS–1	100	50mA per channel
F2–04AD–1L	100	5mA @ 10-30V	F2–02DAS–2	100	60mA per channel
F2–04AD–2	110	5mA @ 10-30V	F2–4AD2DA	90	80mA**
F2–04AD–2L	60	90mA @ 12V**	F2–8AD4DA-1	35	100
F2–08AD–1	100	5mA @ 10-30V	F2–8AD4DA-2	35	80
F2–08AD–2	100	5mA @ 10-30V	F2–04RTD	90	0
F2–02DA–1	40	60**	F2–04THM	110	60
F2–02DA–1L	40	70mA @ 12V**			
F2–02DA–2	40	60			
F2–02DA–2L	40	70mA @ 12V**			
F2–08DA–1	30	50mA**			
F2–08DA–2	60	140			

*requires external 5VDC for outputs

**add an additional 20mA per loop

Power Budget Calculation Example

The following example shows how to calculate the power budget for the DL205 system.

Base # 0	Module Type	5 VDC (mA)	Auxiliary Power Source 24 VDC Output (mA)
Available Base Power	D2–09B–1	2600	300
CPU Slot	D2–260	+ 330	
Slot 0	D2–16ND3–2	+ 100	+ 0
Slot 1	D2–16NA	+ 100	+ 0
Slot 2	D2–16NA	+ 100	+ 0
Slot 3	F2–04AD–1	+ 50	+ 80
Slot 4	F2–02DA–1	+ 40	+ 60
Slot 5	D2–08TA	+ 250	+ 0
Slot 6	D2–08TD1	+ 100	+ 0
Slot 7	D2–08TR	+ 250	+ 0
Other			
Handheld Programmer	D2–HPP	+ 200	+ 0
Total Power Required		**1520**	**140**
Remaining Power Available		2600–1520 = **1080**	300 – 140 = **160**

1. Use the power budget table to fill in the power requirements for all the system components. First, enter the amount of power supplied by the base. Next, list the requirements for the CPU, any I/O modules, and any other devices, such as the Handheld Programmer, C-more HMI or the DV–1000 operator interface. Remember, even though the Handheld Programmer or the DV–1000 are not installed in the base, they still obtain their power from the system. Also, make sure you obtain any external power requirements, such as the 24VDC power required by the analog modules.
2. Add the current columns starting with CPU slot and put the total in the row labeled "Total Power Required."
3. Subtract the row labeled "Total Power Required" from the row labeled "Available Base Power." Place the difference in the row labeled "Remaining Power Available."
4. If "Total Power Required" is greater than the power available from the base, the power budget will be exceeded. It will be unsafe to use this configuration, and you will need to restructure your I/O configuration.

WARNING: It is extremely important to calculate the power budget. If you exceed the power budget, the system may operate in an unpredictable manner which may result in a risk of personal injury or equipment damage.

Power Budget Calculation Worksheet

This blank chart is provided for you to copy and use in your power budget calculations.

Base # 0	Module Type	5 VDC (mA)	Auxiliary Power Source 24 VDC Output (mA)
Available Base Power			
CPU Slot			
Slot 0			
Slot 1			
Slot 2			
Slot 3			
Slot 4			
Slot 5			
Slot 6			
Slot 7			
Other			
Total Power Required			
Remaining Power Available			

1. Use the power budget table to fill in the power requirements for all the system components. This includes the CPU, any I/O modules, and any other devices, such as the Handheld Programmer, C-more HMI or the DV–1000 operator interface. Also, obtain all external power requirements, such as the 24VDC power required by the analog modules.
2. Add the current columns starting with CPU slot and put the total in the row labeled "Total Power Required."
3. Subtract the row labeled "Total Power Required" from the row labeled "Available Base Power." Place the difference in the row labeled "Remaining Power Available."
4. If "Total Power Required" is greater than the power available from the base, the power budget will be exceeded. It will be unsafe to use this configuration, and you will need to restructure your I/O configuration.

WARNING: It is extremely important to calculate the power budget. If you exceed the power budget, the system may operate in an unpredictable manner which could result in personal injury or equipment damage.

Local Expansion I/O

Use local expansion when you need more I/O points, a greater power budget than the local CPU base provides or when placing an I/O base at a location away from the CPU base, but within the expansion cable limits. Each local expansion base requires the D2–CM controller module in the CPU slot. The local CPU base requires the D2–EM expansion module, as well as each expansion base. All bases in the system must be the (–1) bases. These bases have a connector on the right side of the base to which the D2–EM expansion module attaches. All local and local expansion I/O points are updated on every CPU scan.

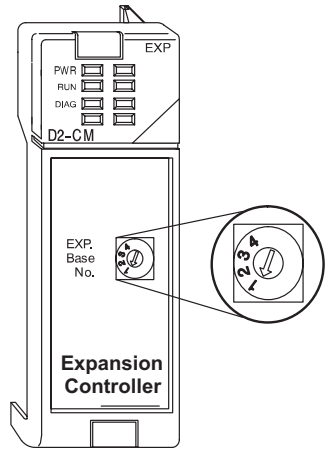
Use the *Direct*SOFT PLC Configure I/O menu option to view the local expansion system automatic I/O addressing configuration. This menu also allows manual addresses to be assigned if necessary.

	D2-230	D2-240	DL250	D2-250-1	D2-260/ D2-262
Total number of local / expansion bases per system	These CPUs do not support local expansion systems			3	5
Maximum number of expansion bases				2	4
Total I/O (includes CPU base and expansion bases)				768	1280
Maximum inputs				512	1024
Maximum outputs				512	1024
Maximum expansion system cable length				30m (98ft.)	

D2–CM Local Expansion Module

The D2–CM module is placed in the expansion base. The rotary switch is used base number. The expansion base I/O ad is based on the numerical order of the r and is recognized by the CPU on expansion base numbers will not be rec

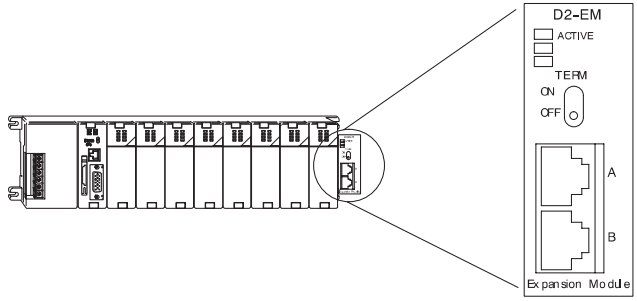
The status indicator LEDs on the panels have specific functions which programming and troubleshooting.



D2–CM Indicators	Status	Meaning
PWR (Green)	ON	Power good
	OFF	Power failure
RUN (Green)	ON	D2–CM has established communication with PLC
	OFF	D2–CM has not established communication with PLC
DIAG (Red)	ON	Hardware watch–dog failure
	ON/OFF	I/O module failure (ON 500ms / OFF 500ms)
	OFF	No D2–CM error

D2–EM Local Expansion Module

The D2–EM expansion unit is attached to the right side of each base in the expansion system, including the local CPU base. (All bases in the local expansion system must be the (–1) bases). The D2–EMs on each end of the expansion system should have the TERM (termination) switch placed in the ON position. The expansion units between the endmost bases should have the TERM switch placed in the OFF position. The CPU base can be located at any base position in the expansion system. The bases are connected in a daisy–chain fashion using the D2–EXCBL–1 (category 5 straight–through cable with RJ45 connectors). Either of the RJ45 ports (labelled A and B) can be used to connect one expansion base to another.



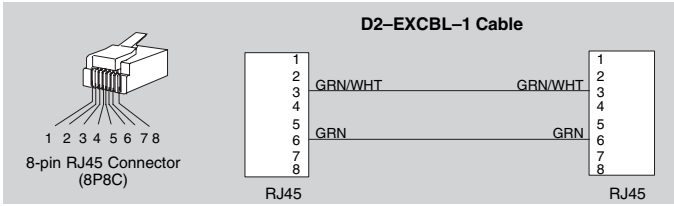
The status indicator LEDs on the D2–EM front panels have specific functions which can help in programming and troubleshooting.

D2–EM Indicator	Status	Meaning
ACTIVE (Green)	ON	D2–EM is communicating with other D2–EM
	OFF	D2–EM is not communicating with other D2–EM

WARNING: Connect/disconnect the expansion cables with the PLC power turned OFF in order for the ACTIVE indicator to function normally.

D2–EXCBL–1 Local Expansion Cable

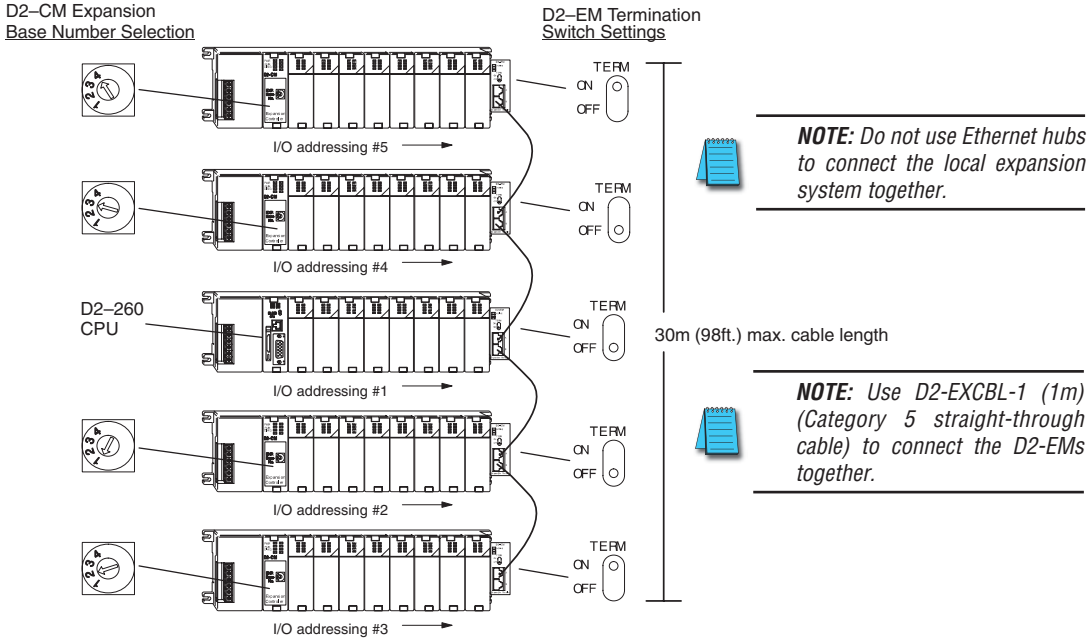
The category 5 straight–through D2–EXCBL–1 (1m) is used to connect the D2–EM expansion modules together. If longer cable lengths are required, we recommend that you purchase a commercially manufactured cable with RJ45 connectors already attached. The maximum total expansion system cable length is 30m (98ft). **Do not use Ethernet hubs** to connect the local expansion network together.



NOTE: *Commercially available Patch (Straight–through) Category 5, UTP cables will work in place of the D2–EXCBL–1. The D2–EM modules only use the wires connected to pins 3 and 6 as shown above.*

D2-260/D2-262 Local Expansion System

The D2-260 and D2-262 support local expansion up to five total bases (one CPU base + four local expansion bases) and up to a maximum of 1280 total I/O points. An example local expansion system is shown below. All local and expansion I/O points are updated on every CPU scan. **No specialty modules can be located in the expansion bases** (refer to the Module Placement Table earlier in this chapter for restrictions).

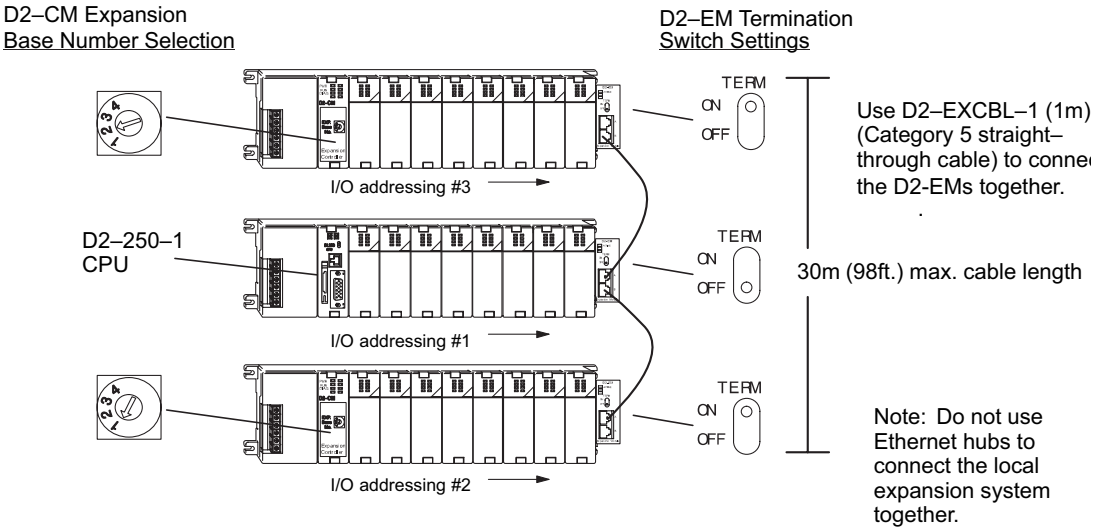


- The CPU base can be located at any base position in the expansion system.
- All discrete and analog modules are supported in the expansion bases. Specialty modules are not supported in the expansion bases.
- The D2–CMs do not have to be in successive numerical order; however, the numerical rotary selection determines the X and Y addressing order. The CPU will recognize the local and expansion I/O on power–up. Do not duplicate numerical selections.
- The TERM (termination) switch on the two endmost D2–EMs must be in the ON position. The other D2–EMs in between should be in the OFF position.
- Use the D2–EXCBL–1 or equivalent cable to connect the D2–EMs together. Either of the RJ45 ports (labeled A and B) on the D2–EM can be used to connect one base to another.

NOTE: *When applying power to the CPU (D2-250-1, D2-260 or D2-262) and local expansion bases, make sure the expansion bases power up at the same time or before the CPU base. Expansion bases that power up after the CPU base will not be recognized by the CPU. (See chapter 3 Initialization Process timing specifications).*

D2-250-1 Local Expansion System

The D2–250–1 supports local expansion up to three total bases (one CPU base + two local expansion bases) and up to a maximum of 768 total I/O points. An example local expansion system is shown below. All local and expansion I/O points are updated on every CPU scan. No specialty modules can be located in the expansion bases (refer to the Module Placement Table earlier in this chapter for restrictions).



- The CPU base can be located at any base position in the expansion system.
- All discrete and analog modules are supported in the expansion bases. Specialty modules are not supported in the expansion bases.
- The D2–CMs do not have to be in successive numerical order, however, the numerical rotary selection determines the X and Y addressing order. The CPU will recognize the local and expansion I/O on power–up. Do not duplicate numerical selections.
- The TERM (termination) switch on the two endmost D2–EMs must be in the ON position. The other D2–EMs in between should be in the OFF position.
- Use the D2–EXCBL–1 or equivalent cable to connect the D2–EMs together. Either of the RJ45 ports (labelled A and B) on the D2–EM can be used to connect one base to another.

Expansion Base Output Hold Option

The bit settings in V–memory registers V7741 and V7742 determine the expansion bases' outputs response to a communications failure. The CPU will exit the RUN mode to the STOP mode when an expansion base communications failure occurs. If the Output Hold bit is ON, the outputs on the corresponding module will hold their last state when a communication error occurs. If OFF (default), the outputs on the module unit will turn off in response to an error. The setting does not have to be the same for all the modules on an expansion base.

The selection of the output mode will depend on your application. You must consider the consequences of turning off all the devices in one or all expansion bases at the same time vs. letting the system run "steady state" while unresponsive to input changes. For example, a conveyor system would typically suffer no harm if the system were shut down all at once. In a way, it is the equivalent of an "E–STOP". On the other hand, for a continuous process such as waste water treatment, holding the last state would allow the current state of the process to continue until the operator can intervene manually. V7741 and V7742 are reserved for the expansion base Output Hold option. The bit definitions are as follows:

Bit = 0 Output Off (Default)

Bit = 1 Output Hold

D2–CM Expansion Base Hold Output										
Expansion Base No.	V–memory Register		Slot 0	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7
Exp. Base 1	V7741	Bit	0	1	2	3	4	5	6	7
Exp. Base 2			8	9	10	11	12	13	14	15
Exp. Base 3	V7742	Bit	0	1	2	3	4	5	6	7
Exp. Base 4			8	9	10	11	12	13	14	15

WARNING: Selecting "HOLD LAST STATE" means that outputs on the expansion bases will not be under program control in the event of a communications failure. Consider the consequences to process operation carefully before selecting this mode.

Enabling I/O Configuration Check using DirectSOFT

Enabling the I/O Config Check will force the CPU, at power up, to examine the local and expansion I/O configuration before entering the RUN mode. If there is a change in the I/O configuration, the CPU will not enter the RUN mode. For example, if local expansion base #1 does not power up with the CPU and the other expansion bases, the I/O Configuration Check will prevent the CPU from entering the RUN mode. If the I/O Configuration check is disabled and automatic addressing is used, the CPU would assign addresses from expansion base #1 to base #2 and possibly enter the RUN mode. This is not desirable, and can be prevented by enabling the I/O Configuration check.

Manual addressing can be used to manually assign addresses to the I/O modules. This will prevent any automatic addressing re–assignments by the CPU. The I/O Configuration Check can also be used with manual addressing.

To display the I/O Config Check window, use ***Direct***SOFT>PLC menu>Setup>I/O Config Check.

Select “Yes,” then save to disk or to PLC, if connected to the PLC.

Expanding DL205 I/O

I/O Expansion Overview

Expanding I/O beyond the local chassis is useful for a system which has a sufficient number of sensors and other field devices located a relatively long distance from the CPU. Two forms of communication can be used to add remote I/O to your system: either an Ethernet or a serial communication network. A discussion of each method follows.

Ethernet Remote Master, H2-ERM(100, -F)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Ethernet Remote Master, H2-ERM(100, -F), is a module that provides a low-cost, high-speed Ethernet Remote I/O link to connect either a D2-240, a D2-250-1, a D2-260 or a D2-262 CPU to slave I/O over a high-speed Ethernet link.

Each H2-ERM(100) module can support up to 16 additional H2-EBC(100) systems, 16 Terminator I/O EBC systems, or 16 fully expanded H4-EBC systems.

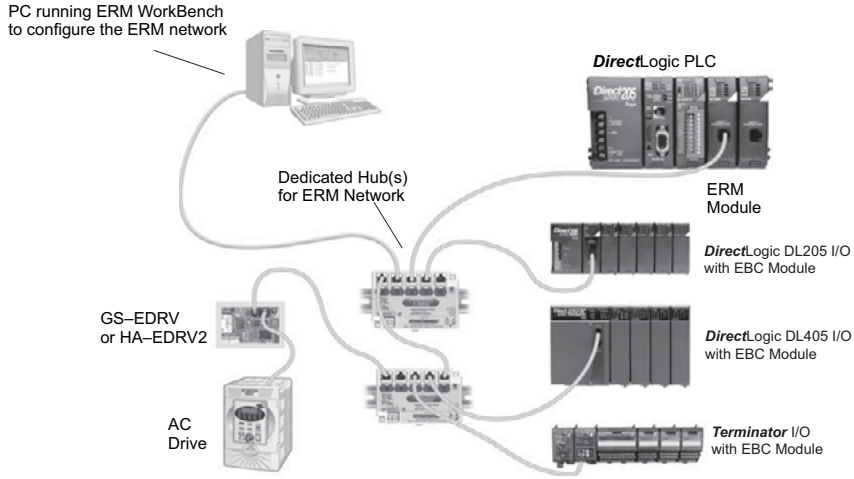
The H2-ERM(100) connects to your control network using Category 5 UTP cables for distances up to 100m (328ft). Repeaters are used to extend the distances and to expand the number of nodes. The fiber optic version, H2-ERM-F, uses industry standard 62.5/125 ST-style fiber optic cables and can be run up to 2000m (6560ft).

The PLC, ERM and EBC slave modules work together to update the remote I/O points. These three scan cycles are occurring at the same time, but asynchronously. We recommend that critical I/O points that must be monitored every scan be placed in the CPU base.

Specifications	H2-ERM	H2-ERM100	H2-ERM-F
Communications	10BaseT Ethernet	10/100BaseT Ethernet	10BaseFL Ethernet
Data Transfer Rate	10Mbps	100Mbps	10Mbps
Link Distance	100 meters (328ft)		2000 meters (6560ft)
Ethernet Port	RJ45		ST-style fiber optic
Ethernet Protocols	TCP/IP, IPX	TCP/IP, IPX, Modbus TCP/IP, DHCP, HTML configuration	TCP/IP, IPX
Power Consumption	320mA @ 5VDC		450mA @ 5VDC

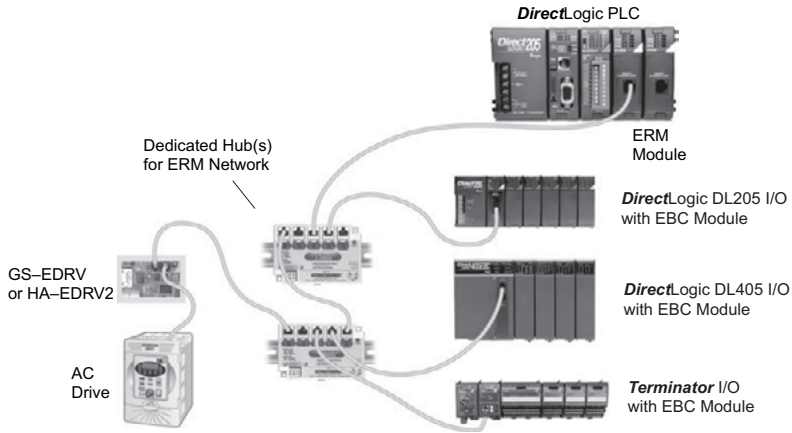
Ethernet Remote Master Hardware Configuration

Use a PC equipped with a 10/100BaseT or a 10BaseFL network adapter card and the Ethernet Remote Master (ERM) Workbench software configuration utility (the ERM Workbench software and the ERM manual, H24-ERM-M, are both available for download on the **AutomationDirect.com** website) to configure the ERM module and its slaves over the Ethernet remote I/O network.



When networking ERMs with other Ethernet devices, we recommend that a dedicated Ethernet remote I/O network be used for the ERM and its slaves. While Ethernet networks can handle an extremely large number of data transactions, and normally very quickly, heavy Ethernet traffic can adversely affect the reliability of the slave I/O and the speed of the I/O network. Keep ERM networks, multiple ERM networks and ECOM/office networks isolated from one another.

Once the ERM remote I/O network is configured and running, the PC can be removed from the network.



Installing the ERM Module

This section will briefly describe the installation of the ERM module. More detailed information is available in the Ethernet Remote Master Module manual, H24-ERM-M, which will be needed to configure the communication link to the remote I/O.

In addition to the manual, configuration software will be needed. The ERM Workbench software utility must be used to configure the ERM and its slave modules. The ERM user manual, H24-ERM-M and the ERM Workbench utility are available for download at the **AutomationDirect.com** website. The ERM module can be identified by two different methods, either by Module ID (dip switch) or by Ethernet address. Whichever method is used, the ERM Workbench is all that is needed to configure the network modules.

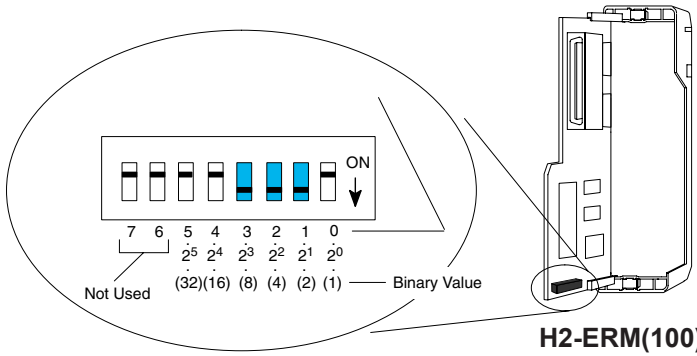
If IP addressing (UDP/IP) is necessary or if the Module ID is set with software, the NetEdit software utility (included with the ERM Workbench utility) will be needed in addition to the ERM Workbench.

ERM Module ID

Set the ERM Module ID before installing the module in the DL205 base. Always set the module ID to 0. A Module ID can be set in one of two ways:

- Use the DIP switches on the module (1-63)
- Use the configuration tools in NetEdit

Use the DIP switch to install and change slave modules without using a PC to set the Module ID. Set the module's DIP switch, insert the module in the base, and connect the network cable. The Module ID is set on power up, and it is ready to communicate on the network.

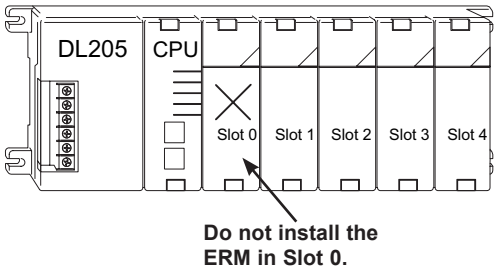


The Module IDs can also be set or changed on the network from a single PC by using the tools in NetEdit.

The Module ID equals the *sum* of the binary values of the slide switches set in the ON position. For example, if slide switches 1, 2 and 3 are set to the ON position, the Module ID will be 14. This is found by adding 8+4+2=14. The maximum value which can be set on the DIP switch is 32+16+8+4+2=63. This is achieved by setting switches 0 through 5 to the ON position. The 6 and 7 switch positions are inactive.

Insert the ERM Module

The DL205 system ***only*** supports the placement of the ERM module in the **CPU base**. It does not support installation of the ERM module in either local expansion or remote I/O bases. The number of usable slots depends on how many slots the base has. All of the DL205 CPUs support the ERM module, except the D2-230.



NOTE: *The module will not work in slot 0 of the DL205 series PLCs, the slot next to the CPU.*

Network Cabling

Of the three types of ERM modules available, one supports the 10BaseT standard, another supports 10/100BaseT and the other one supports the 10BaseFL standard. The 10/100BaseT standard uses twisted pairs of copper wire conductors and the 10BaseFL standard is used with fiber optic cabling.

10/100BaseT

Unshielded
Twisted-Pair
cable with RJ45
connectors

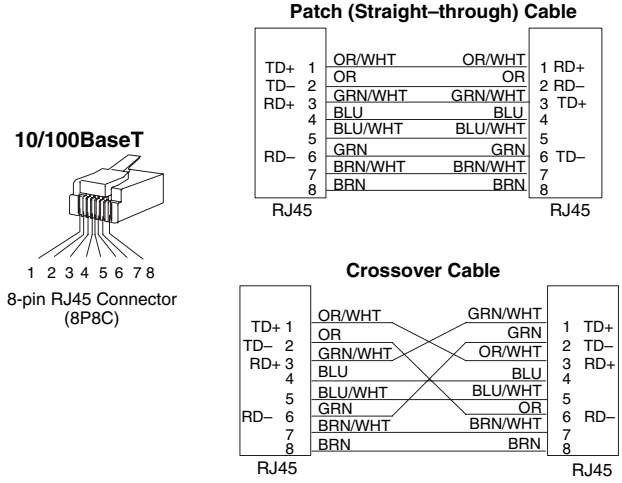
10BaseFL

62.5/125 MMF
fiber optics cable
with ST-style
connectors

10/100BaseT Networks

A patch (straight-through) cable is used to connect a PLC (or PC) to a hub or to a repeater. Use a crossover cable to connect two Ethernet devices (point-to-point) together. It is recommended that pre-assembled cables be purchased for convenient and reliable networking.

The above diagram illustrates the standard wire positions of the RJ45 connector. It is recommended that Catagory 5, UTP cable be used for all ERM 10/100BaseT cables.



Refer to the ERM manual for using the fiber optic cable with the H2-ERM-F.

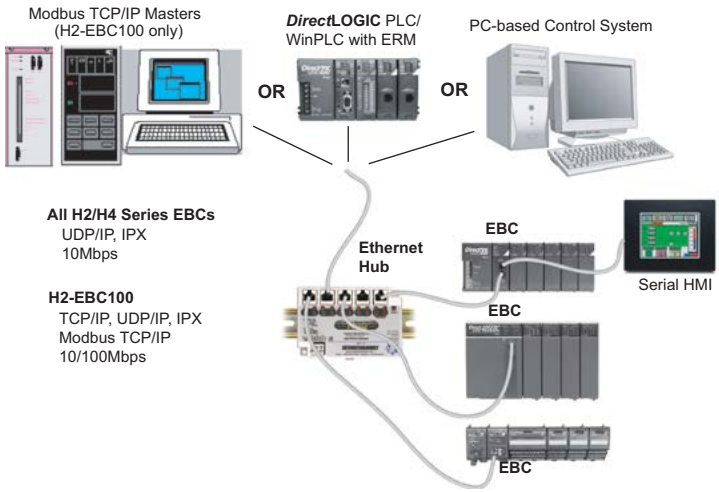
An explanation of the use of the ERM Workbench software is too lengthy for this manual. The full use of the workbench and NetEdit utilities is discussed in the ERM manual.

Ethernet Base Controller, H2-EBC(100)(-F)

The Ethernet Base Controller module H2-EBC(100)(-F) provides a low-cost, high-performance Ethernet link between a network master controller and an *Direct*LOGIC PLC I/O slave system. Also, the H2-EBC100 supports the Modbus TCP/IP server protocol.

The Ethernet Base Controller (EBC) serves as an interface between the master control system and the DL205/405 I/O modules. The control function is performed by the master controller, not the EBC slave. The EBC occupies the CPU slot in the base and communicates across the backplane to input and output modules. Various master controllers with EBC slaves are shown in the diagram below.

Example EBC Systems: Various Masters with EBC Slaves



The H2-EBC module supports industry standard 10BaseT Ethernet communications, the H2-EBC100 module supports industry standard 10/100BaseT Ethernet communications and the H2-EBC-F module supports 10BaseFL (fiber optic) Ethernet standards.

Specifications	H2-EBC	H2-EBC100	H2-EBC-F
Communications	10BaseT Ethernet	10/100BaseT Ethernet	10BaseFL Ethernet
Data Transfer Rate	10 Mbps max.	100 Mbps max.	10 Mbps max.
Link Distance	100m (328ft)	100m (328ft)	2000m (6560ft)
Ethernet Port	RJ45	RJ45	ST-style fiber optic
Ethernet Protocols	TCP/IP, IPX	TCP/IP, IPX/Modbus TCP/IP, DHCP, HTML configuration	TCP/IP, IPX
Serial Port	RJ12	RJ12	None
Serial Protocols*	K-Sequence, ASCII IN/ OUT	K-Sequence, ASCII IN/OUT, Modbus RTU	None
Power Consumption	450mA @ 5VDC	300mA @ 5VDC	640mA @ 5VDC

* Serial communications support available with PC based control master (Think n Do). It does not support HMI communications.

Install the EBC Module

Like the ERM module discussed in the previous section, this section will briefly describe the installation of the H2 Series EBCs. More detailed information is available in the Ethernet Base Controller manual, H24-EBC-M, which will be needed to configure the remote I/O.

Each EBC module must be assigned at least one unique identifier to make it possible for master controllers to recognize it on the network. Two methods for identifying the EBC module give it the flexibility to fit most networking schemes. These identifiers are:

- Module ID (IPX protocol only)
- IP Address (for TCP/IP and Modbus TCP/IP protocols)

Set the Module ID

The two methods which can be used to set the EBC module ID are either by DIP switch or by software. One software method is to use the NetEdit3 program which is is installed with DirectSOFT or available for download at **AutomationDirect.com** website. To keep the set-up discussion simple here, only the DIP switch method will be discussed. Refer to the EBC manual for the complete use of NetEdit3.

It is recommended to use the DIP switch to set the Module ID because the DIP switch is simple to set, and the Module ID can be determined by looking at the physical module, without reference to a software utility.

The DIP switch can be used to set the Module ID to a number from 1-63. Do not use Module ID 0 for communication.

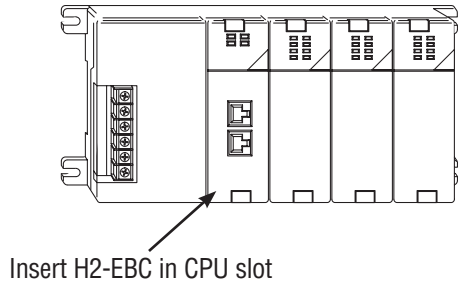
If the DIP switch is set to a number greater than 0, the software utilities are disabled from setting the Module ID. Software utilities will only allow changes to the Module ID if the DIP switch setting is 0 (all switches OFF).

NOTE: *The DIP switch settings are read at powerup only. The power must be cycled each time the DIP switches are changed.*

Setting the Module ID with the DIP switches is identical to setting the DIP switches on the H2-ERM(100) module. Refer to page 4-19 in this chapter.

Insert the EBC Module

Once the Module ID DIP switches are set, insert the module in the CPU slot of any DL205 base.



Network Cabling

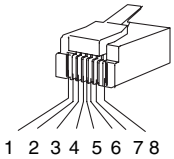
Of the two types of EBC modules available, one supports the 10/100BaseT standard and the other one supports the 10BaseFL standard. The 10/100BaseT standard uses twisted pairs of copper wire conductors and the 10BaseFL standard is used with fiber optic cabling.

10/100BaseT

The 10BaseT and 100BaseT EBCs have an eight-pin modular jack that accepts RJ45 connectors. UTP Category 5 (CAT5) cable is highly recommended for use with all Ethernet 10/100BaseT connections. For convenient and reliable networking, purchase commercially manufactured cables which have the connectors already installed.

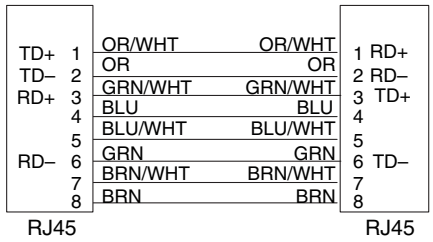
To connect an EBC, or a PC, to a hub or repeater, use a patch cable (sometimes called a straight-through cable). The cable used to connect a PC directly to an EBC or to connect two hubs is referred to as a crossover cable.

10/100BaseT

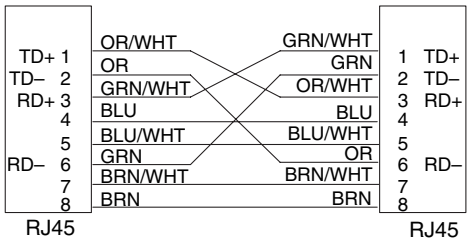


8-pin RJ45 Connector
(8P8C)

Patch (Straight–through) Cable



Crossover Cable

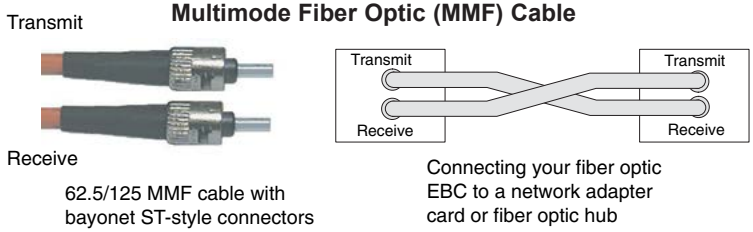


10BaseFL Network Cabling

The H2-EBC-F and the H2-ERM-F modules have two ST-style bayonet connectors. The ST-style connector uses a quick release coupling which requires a quarter turn to engage or disengage. The connectors provide mechanical and optical alignment of fibers.

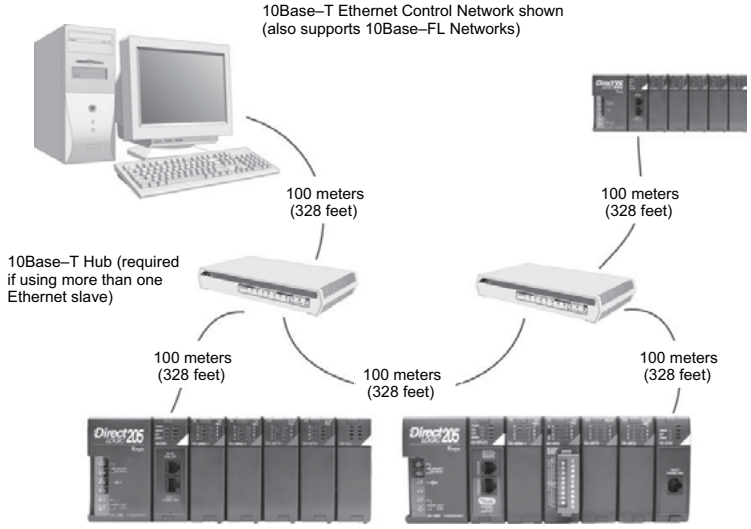
Each cable segment requires two strands of fiber; one to transmit data and one to receive data. The ST-style connectors are used to connect the H2-Exx-F module to a PC or a fiber optic hub or repeater. The modules themselves cannot act as repeaters.

The H2-EBC-F and the H2-ERM-F modules accept 62.5/125 multimode fiber optic (MMF) cable. The glass core diameter is 62.5 micrometers, and the glass cladding is 125 micrometers. The fiber optic cable is highly immune to noise and permits communications over much greater distances than 10/100BaseT.



Maximum Cable Length

The maximum distance per 10/100BaseT cable segment is 100 meters (328 feet). Repeaters extend the distance. Each cable segment attached to a repeater can be 100 meters. Two repeaters connected together extend the total range to 300 meters. The maximum distance per 10BaseFL cable segment is 2,000 meters (6,560 feet or 1.2 miles). Repeaters extend the distance. Each cable segment attached to a repeater can be 2,000 meters. Two repeaters connected together extend the total range to 6,000 meters.



Add a Serial Remote I/O Master/Slave Module (No longer available for new applications)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

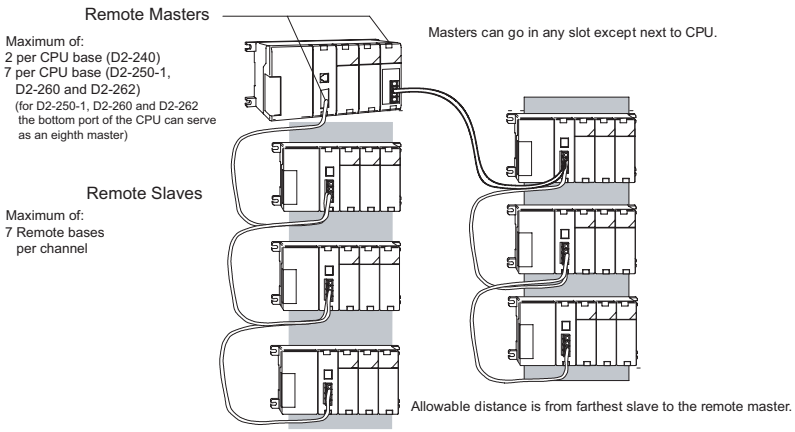
In addition to the I/O located in the local base, adding remote I/O can be accomplished via a shielded twisted-pair cable linking the master CPU to a remote I/O base. The methods of adding serial remote I/O are:

- D2-240 CPUs: Remote I/O requires a remote master module (D2–RMSM) to be installed in the local base. The CPU updates the remote master, then the remote master handles all communication to and from the remote I/O base by communicating to a remote slave module (D2–RSSS) installed in each remote base.
- D2-250-1, D2–260 and D2-262 CPU: The CPU comm port 2, features a built-in Remote I/O channel. You may also use up to seven D2–RMSM remote masters in the local base as described above (you can use either or both methods).

	D2-230	D2-240	D2-250-1	D2-260/ D2-262
Maximum number of Remote Masters supported in the local CPU base (1 channel per Remote Master)	None	2	7	7
CPU built-in Remote I/O channels	None	None	1	1
Maximum I/O points supported by each channel	None	2048	2048	2048
Maximum Remote I/O points supported	None	Limited by total references available		
Maximum number of Remote I/O bases per channel(RM–NET)	None	7	7	7
Maximum number of Remote I/O bases per channel (SM–NET)	None	31	31	31

Remote I/O points map into different CPU memory locations, therefore it does not reduce the number of local I/O points. Refer to the DL205 Remote I/O manual for details on remote I/O configuration and numbering. Configuring the built-in remote I/O channel is described in the following section.

The figure below shows one CPU base, and one remote I/O channel with six remote bases. If the CPU is a D2-250-1, D2-260 or D2-262, adding the first remote I/O channel does not require installing a remote master module (use the CPU's built-in remote I/O channel).



Configuring the CPU's Remote I/O Channel

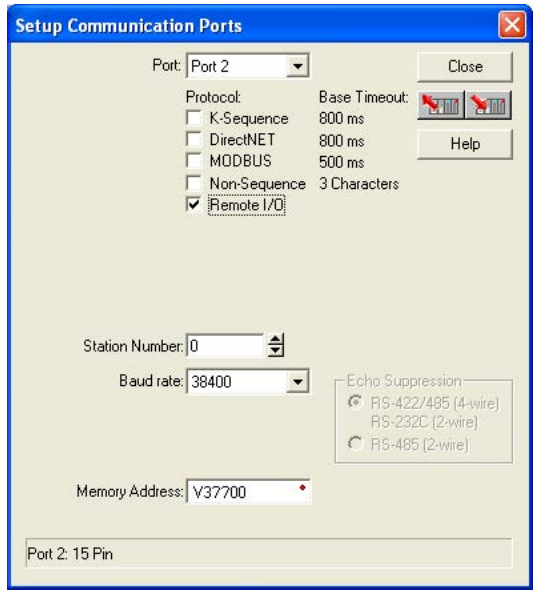
☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

This section describes how to configure the D2-250-1, D2-260 and D2-262 CPU built-in remote I/O channel. Additional information is in the Remote I/O manual, D2–REMIO–M, which you will need in configuring the Remote slave units on the network. You can use the D2–REMIO–M manual exclusively when using regular Remote Masters and Remote Slaves for remote I/O in any DL205 system.

The D2-250-1, D2-260 and D2-262 CPUs have a built-in remote I/O channel which supports RM–Net allowing it to communicate with up to seven remote bases containing a maximum of 2048 I/O points per channel, at a maximum distance of 1000 meters. If required, you can still use Remote Master modules in the local CPU base (2048 I/O points on each channel).

You may recall from the CPU specifications in Chapter 3 that the D2-250–1, D2-260 and D2-262's Port 2 is capable of several protocols. To configure the port using the Handheld Programmer, use AUX 56 and follow the prompts, making the same choices as indicated below on this page. To configure the port in *Direct*SOFT, choose the PLC menu, then Setup, then Setup Secondary Comm Port.

- Port: From the port number list box at the top, choose "Port 2."
- Protocol: Click the check box to the left of "Remote I/O" (called "M–NET" on the HPP), and then you'll see the dialog box shown below.
- Station Number: Choose "0" as the station number, which makes the D2-250-1, D2-260 or D2-262 CPU the master. Station numbers 1–7 are reserved for remote slaves.
- Baud Rate: The baud rates 19200 and 38400 are available. Choose 38400 initially as the remote I/O baud rate, and revert to 19200 baud if you experience data errors or noise problems on the link.
- Memory Address: Choose a V-memory address to use as the starting location of a Remote I/O configuration table (V37700 is the default). This table is separate and independent from the table for any Remote Master(s) in the system, and it is 32 words in length.

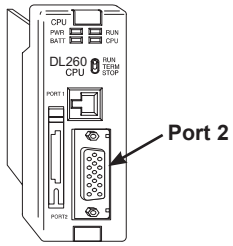


Then click the button indicated to send the Port 2 configuration to the CPU, and click Close.

NOTE: *You must configure the baud rate on the Remote Slaves with DIP switches to match the baud rate selection for the CPU's Port 2.*

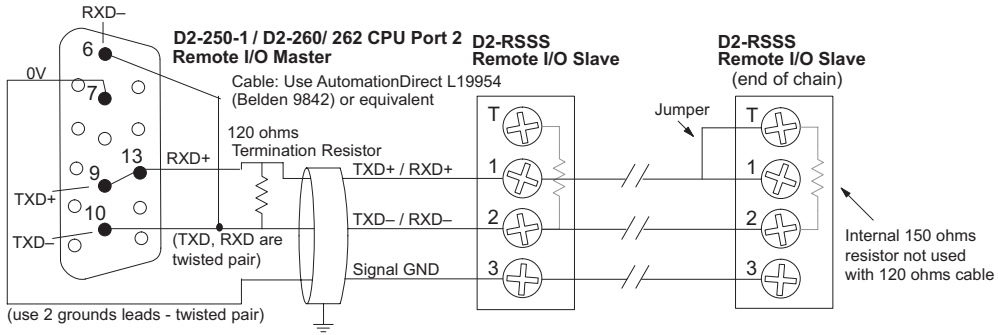
The next step is to make the connections between all devices on the Remote I/O link.

Port 2 location for the D2-250-1, D2-260 and D2-262 is the 15-pin connector, as pictured to the right.



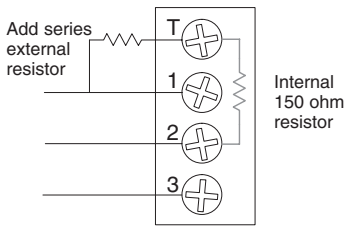
- Pin 7 Signal GND
- Pin 9 TXD+
- Pin 10 TXD–
- Pin 13 RXD+
- Pin 6 RXD–

Now we are ready to discuss wiring the D2-250-1, D2-260 or D2-262 to the remote slaves on the remote base(s). The remote I/O link is a 3-wire, half-duplex type. Since Port 2 of the D2-250–1, D2-260 and D2-262 CPU is a 5-wire full duplex–capable port, we must jumper its transmit and receive lines together as shown below (converts it to 3-wire, half-duplex).



The twisted/shielded pair connects to the D2-250-1, D2-260 or D2-262 Port 2 as shown. A termination resistor must be added externally to the CPU, as close as possible to the connector pins. Its purpose is to minimize electrical reflections that occur over long cables. A termination resistor must be present at both physical ends of the network.

Ideally, the two termination resistors at the cable's opposite ends and the cable's rated impedance will all match. For cable impedances greater than 150 ohms, add a series resistor at the last slave as shown to the right. If less than 150 ohms, parallel a matching resistance across the slave's pins 1 and 2 instead. Remember to size the termination resistor at Port 2 to match the cables rated impedance. ***The resistance values should be between 100 and 500 ohms.***



NOTE: *To match termination resistance to AutomationDirect L19827 (Belden 9841), use a 120 ohm resistor across terminals 1 and 2.*

NOTE: *See the transient suppression for inductive loads information in Chapter 2 of this manual for further information on wiring practices.*

Configure Remote I/O Slaves

After configuring either the D2-250-1, D2-260 or D2-262 CPU Port 2 and wiring it to the remote slave(s), use the following checklist to complete the configuration of the remote slave(s). Full instructions for these steps are in the Remote I/O manual.

- Set the baud rate to match CPU Port 2 setting.
- Select a station address for each slave, from 1 to 7. Each device on the remote link must have a unique station address. There can be only one master (address 0) on the remote link.

Configuring the Remote I/O Table

The beginning of the configuration table for the built-in remote I/O channel is the memory address we selected in the Port 2 setup.

The table consists of blocks of four words which correspond to each slave in the system, as shown to the right. The first four table locations are reserved.

The CPU reads data from the table after powerup, interpreting the four data words in each block with these meanings:

1. Starting address of slave input data
2. Number of slave input points
3. Starting address of outputs in slave
4. Number of slave output points

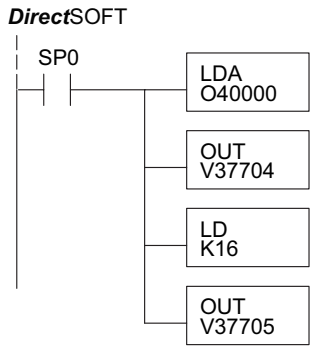
The table is 32 words long. If your system has fewer than seven remote slave bases, then the remainder of the table must be filled with zeros. For example, a three–slave system will have a remote configuration table containing four reserved words, 12 words of data and 16 words of "0000."

A portion of the ladder program must configure this table (only once) at powerup. Use the LDA instruction as shown to the right, to load an address to place in the table. Use the regular LD constant to load the number of the slave's input or output points. The following page gives a short program example for one slave.

Memory Addr. Pointer	37700

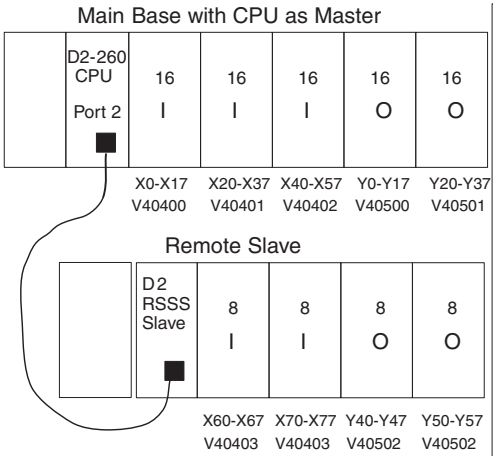
Remote I/O data

Reserved	V37700	xxxx
	V37701	xxxx
	V37702	xxxx
	V37703	xxxx
Slave 1	V37704	xxxx
	V37705	xxxx
	V37706	xxxx
	V37707	xxxx
Slave 7	V37734	0000
	V37735	0000
	V37736	0000
	V37737	0000



Consider the simple system featuring Remote I/O shown below. The D2-250-1, D2-260 or D2-262's built-in Remote I/O channel connects to one slave base, which we will assign a station address=1. The baud rates on the master and slave will be 38.4KB.

We can map the remote I/O points as any type of I/O point, simply by choosing the appropriate range of V-memory. Since we have plenty of standard I/O addresses available (X and Y), we will have the remote I/O points start at the next X and Y addresses after the main base points (X60 and Y40, respectively).



Remote Slave Worksheet

Remote Base Address ___1___(Choose 1–7)

Slot Number	Module Name	INPUT		OUTPUT	
		Input A d d r	No. Inputs	Output Addr.	No.Outputs
0	08ND3S	X060	8		
1	08ND3S	X070	8		
2	08TD1			Y040	8
3	08TD1			Y050	8
4					
5					
6					
7					

Input Bit Start Address: _ _ _X060 V-Memory Address:V _40403_ _

To ta l In p u t P o in ts16_ _

Output Bit Start Address: _ _ _Y040 V-Memory Address:V _40502_ _

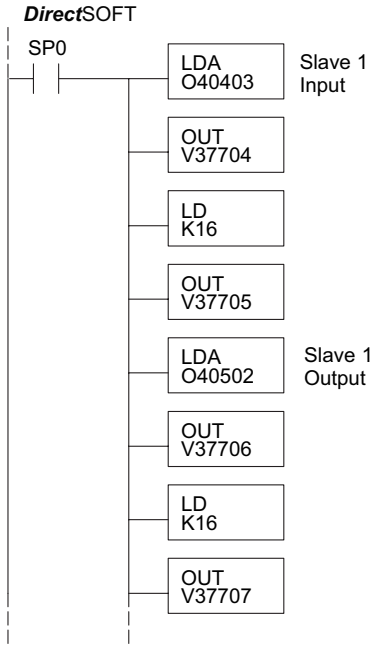
Total Output Points _ _16_ _

Remote I/O Setup Program

Using the Remote Slave Worksheet shown above can help organize our system data in preparation for writing our ladder program (a blank full-page copy of this worksheet is in the Remote I/O Manual). The four key parameters we need to place in our Remote I/O configuration table are in the lower right corner of the worksheet. You can determine the address values by using the memory map given at the end of Chapter 3, CPU Specifications and Operation.

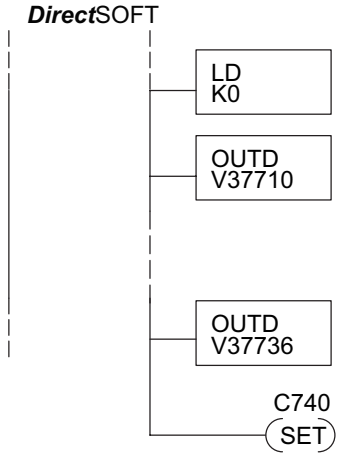
The program segment required to transfer our worksheet results to the Remote I/O configuration table is shown to the right. Remember to use the LDA or LD instructions appropriately.

The next page covers the remainder of the required program to get this remote I/O link up and running.



When configuring a Remote I/O channel for fewer than 7 slaves, we must fill the remainder of the table with zeros. This is necessary because the CPU will try to interpret any non-zero number as slave information.

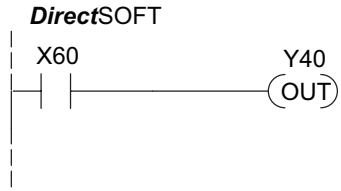
We continue our set-up program from the previous page by adding a segment which fills the remainder of the table with zeros. The example to the right fills zeros for slave numbers 2–7, which do not exist in our example system.



On the last rung in the example program above, we set a special relay contact C740. This particular contact indicates to the CPU the ladder program has finished specifying a remote I/O system. At that moment, the CPU begins remote I/O communications. Be sure to include this contact after any Remote I/O set-up program.

Remote I/O Test Program

Now we can verify the remote I/O link and set-up program operation. A simple quick check can be done with one rung of ladder, shown to the right. It connects the first input of the remote base with the first output. After placing the PLC in RUN mode, we can go to the remote base and activate its first input. Then its first output should turn on.



Network Connections to Modbus and DirectNET

Configuring Port 2 For DirectNET

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

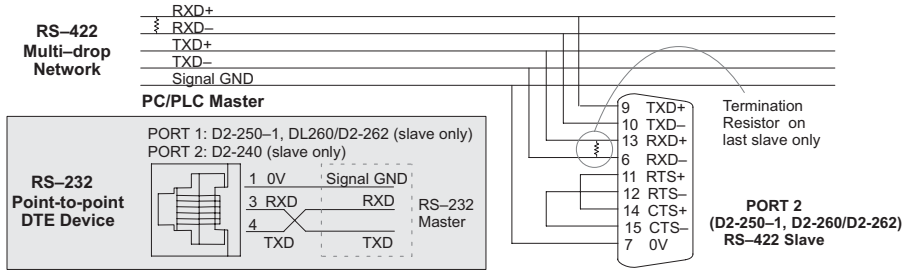
This section describes how to configure the CPU's built-in networking ports for either Modbus or *Direct*NET. This will allow you to connect the DL205 PLC system directly to Modbus networks using the RTU protocol, or to other devices on a *Direct*NET network. For more details on *Direct*NET, order our *Direct*NET manual, part number DA–DNET–M.

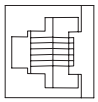
Configuring Port 2 For Modbus RTU

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

Modbus hosts system on the network must be capable of issuing the Modbus commands to read or write the appropriate data. For details on the Modbus protocol, refer to the Modicon Modbus Protocol Reference Guide, PI-MBUS-300, found at Modbus.org. In the event a more recent version is available, check with your Modbus supplier before ordering the documentation.

You will need to determine whether the network connection is a 3-wire RS–232 type, or a 5-wire RS–422 type. Normally, the RS–232 signals are used for shorter distance (15 meters (50 feet) maximum) communications between two devices. RS–422 signals are for longer distance (1000 meters (3280ft) maximum) multi-drop networks (from two to 247 devices). Use termination resistors at both ends of RS–422 network wiring, matching the impedance rating of the cable (between 100 and 500 ohms).

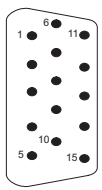




6-pin Female
Modular Connector

Port 1 Pinouts (D2-250–1 / D2-260*)		
1	0V	Power (–) connection (GND)
2	5V	Power (+) conection
3	RXD	Receive Data (RS-232)
4	TXD	Transmit Data (RS-232)
5	5V	Power (+) conection
6	0V	Power (–) connection (GND)

Port 2 Pin Descriptions (D2-240 only)		
1	0V	Power (–) connection (GND)
2	5V	Power (+) conection
3	RXD	Receive Data (RS-232)
4	TXD	Transmit Data (RS-232)
5	RTS	Request to Send
6	0V	Power (–) connection (GND)



15-pin Female
D-Sub connector

Port 2 Pin Descriptions (D2-250–1 / D2-260*)		
1	5V	5 VDC
2	TXD2	Transmit Data (RS-232)
3	RXD2	Receive Data (RS-232)
4	RTS2	Ready to Send (RS–232)
5	CTS2	Clear to Send (RS–232)
6	RXD2–	Receive Data – (RS–422) (RS–485 D2-260)
7	0V	Logic Ground
8	0V	Logic Ground
9	TXD2+	Transmit Data + (RS–422) (RS–485 D2-260)
10	TXD2 –	Transmit Data – (RS–422) (RS–485 D2-260)
11	RTS2 +	Request to Send + (RS–422) (RS–485 D2-260)
12	RTS2 –	Request to Send – (RS–422)(RS–485 D2-260)
13	RXD2 +	Receive Data + (RS–422) (RS–485 D2-260)
14	CTS2 +	Clear to Send + (RS422) (RS–485 D2-260)
15	CTS2 –	Clear to Send – (RS–422) (RS–485 D2-260)

* Applies to D2-262 CPU also.

The recommended cable for RS-232 or RS-422 is AutomationDirect L19772 (Belden 8102) or equivalent. The recommended cable for RS-485 is AutomationDirect L19827 (Belden 9841) or equivalent.

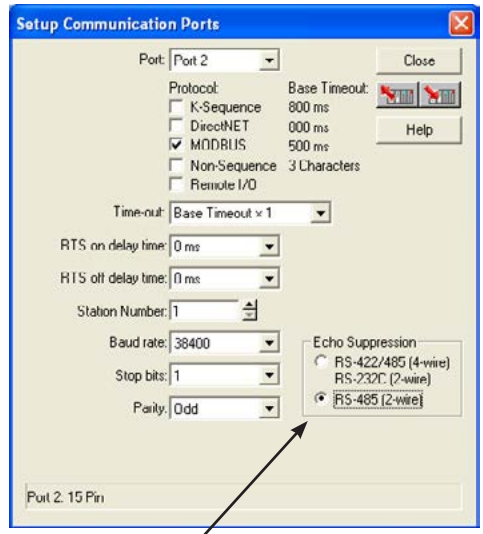
Note: The D2-260/D2-262 supports RS–485 multi–drop networking. See the Network Master Operation (D2-260/D2-262 only) section later in this chapter for details.

Modbus Port Configuration

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

In *Direct*SOFT, choose the PLC menu, then Setup, then "Secondary Comm Port."

- **Port:** From the port number list box at the top, choose "Port 2."
- **Protocol:** Click the check box to the left of "MODBUS" (use AUX 56 on the HPP, and select "MBUS"), and then you'll see the dialog box below.



- **Timeout:** The amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS On Delay Time:** The amount of time between raising the RTS line and sending the data.
- **RTS Off Delay Time:** The amount of time between resetting the RTS line after sending the data.
- **Station Number:** To make the CPU port a Modbus master, choose "1." The possible range for Modbus slave numbers is from 1 to 247, but the D2-250–1, D2-260 and D2-262 WX and RX network instructions used in Master mode will access only slaves 1 to 90. Each slave must have a unique number. At powerup, the port is automatically a slave, unless and until the D2-250-1, D2-260 or D2-262 executes ladder logic network instructions which use the port as a master. Thereafter, the port reverts back to slave mode until ladder logic uses the port again.

***NOTE:** The D2-250-1 does not support the Echo Suppression feature*

- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value. Refer to the appropriate product manual for details.
- **Stop Bits:** Choose 1 or 2 stop bits for use in the protocol.
- **Parity:** Choose none, even, or odd parity for error checking.
- **Echo Suppression:** Select the appropriate radio button based on the wiring configuration used on port 2.

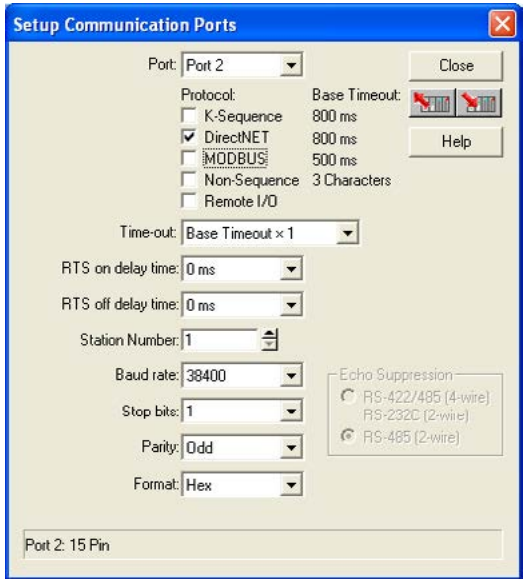
Then click the button indicated to send the Port configuration to the CPU, and click Close.

DirectNET Port Configuration

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

In *Direct*SOFT, choose the PLC menu, then Setup, then "Secondary Comm Port."

- **Port:** From the port number list box, choose "Port 2."
- **Protocol:** Click the check box to the left of "*Direct*NET" (use AUX 56 on the HPP, then select "DNET"), and then you'll see the dialog box below.



- **Timeout:** The amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS On Delay Time:** The amount of time between raising the RTS line and sending the data.
- **RTS Off Delay Time:** The amount of time between resetting the RTS line after sending the data.
- **Station Number:** To make the CPU port a *Direct*NET master, choose "1". The allowable range for *Direct*NET slaves is from 1 to 90 (each slave must have a unique number). At powerup, the port is automatically a slave, unless and until the D2-250-1, D2-260 or D2-262 executes ladder logic instructions which attempt to use the port as a master. Thereafter, the port reverts back to slave mode until ladder logic uses the port again.
- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value.
- **Stop Bits:** Choose 1 or 2 stop bits for use in the protocol.
- **Parity:** Choose none, even, or odd parity for error checking.
- **Format:** Choose hex or ASCII formats.

Then click the button indicated to send the Port configuration to the CPU, and click Close.

Network Slave Operation

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

This section describes how other devices on a network can communicate with a CPU port that you have configured as a *Direct*NET slave (D2-240, 250–1, D2-260 and D2-262) or Modbus slave (D2-250–1, D2-260 and D2-262). A Modbus host must use the Modbus RTU protocol to communicate with the D2-250-1, D2-260 or D2-262 as a slave. The host software must send a Modbus function code and Modbus address to specify a PLC memory location the D2-250-1, D2-260 or D2-262 comprehends. The *Direct*NET host uses normal I/O addresses to access applicable DL205 CPU and system information. No CPU ladder logic is required to support either Modbus slave or *Direct*NET slave operation.

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

Modbus Function Codes Supported

The Modbus function code determines whether the access is a read or a write, and whether to access a single data point or a group of them. The D2-250–1, D2-260 and D2-262 support the Modbus function codes described below.

Modbus Function Code	Function	DL205 Data Types Available
01	Read a group of coils	Y, C, T, CT
02	Read a group of inputs	X, SP
05	Set / Reset a single coil (slave only)	Y, C, T, CT
15	Set / Reset a group of coils	Y, C, T, CT
03, 04	Read a value from one or more registers	V
06	Write a value into a single register (slave only)	V
16	Write a value into a group of registers	V

Determining the Modbus Address

There are typically two ways that most host software conventions allow you to specify a PLC memory location. These are:

- By specifying the Modbus data type and address
- By specifying a Modbus address only.

If Your Host Software Requires the Data Type and Address

Many Host software packages allow you to specify the Modbus data type and the Modbus address that correspond to the PLC memory location. This is the easiest method, but not all packages allow you to do it this way.

The actual equation used to calculate the address depends on the type of PLC data you are using. The PLC memory types are split into two categories for this purpose.

- Discrete – X, SP, Y, C, S, T (contacts), CT (contacts)
- Word – V, Timer current value, Counter current value

In either case, you basically convert the PLC octal address to decimal and add the appropriate Modbus address (if required). The table on the following page shows the exact equation used for each group of data.

NOTE: *For information about the Modbus protocol see www.Modbus.org and select Technical Resources. For more information about the DirectNET protocol, download the DirectNET User Manual, DA-DNET-M, for free from our website: www.automationdirect.com. Select Manuals/Docs>Online User Manuals>Misc.> DA-DNET-M*

D2-250-1 Memory Type	QTY (Dec.)	PLC Range (Octal)	Modbus Address Range (Decimal)	Modbus Data Type
For Discrete Data Types ..... Convert PLC Addr. to Dec. +			**Start of Range +**	**Data Type**
Inputs (X)	512	X0 – X777	2048 – 2560	Input
Special Relays (SP)	512	SP0 – SP137 SP320 – SP717	3072 – 3167 3280 – 3535	Input
Outputs (Y)	512	Y0 – Y777	2048 – 2560	Coil
Control Relays (C)	1024	C0 – C1777	3072 – 4095	Coil
Timer Contacts (T)	256	T0 – T377	6144 – 6399	Coil
Counter Contacts (CT)	128	CT0 – CT177	6400 – 6527	Coil
Stage Status Bits (S)	1024	S0 – S1777	5120 – 6143	Coil
For Word Data Types ..... Convert PLC Addr. to Dec. +				**Data Type**
Timer Current Values (V)	256	V0 – V377	0 – 255	Input Register
Counter Current Values (V)	128	V1000 – V1177	512 – 639	Input Register
V-Memory, user data (V)	3072 4096	V1400 – V7377 V10000 – V17777	768 – 3839 4096 – 8191	Holding Register
V-Memory, system (V)	256	V7400 – V7777	3480 – 3735	Holding Register

D2-260/D2-262 Memory Type	QTY (Dec.)	PLC Range (Octal)	Modbus Address Range (Decimal)	Modbus Data Type
For Discrete Data Types ..... Convert PLC Addr. to Dec. +			**Start of Range +**	**Data Type**
Inputs (X)	1024	X0 – X1777	2048 – 3071	Input
Remote Inputs (GX)	2048	GX0 – GX3777	3840 – 18431	Input
Special Relays (SP)	512	SP0 – SP777	3072 – 3583	Input
Outputs (Y)	1024	Y0 – Y777	2048 – 3071	Coil
Remote Outputs (GY)	2048	GY0 – GY3777	18432 – 20479	Coil
Control Relays (C)	2048	C0 – C377	3072 – 5159	Coil
Timer Contacts (T)	256	T0 – T177	6144 – 6399	Coil
Counter Contacts (CT)	256	CT0 – CT177	6400 – 6655	Coil
Stage Status Bits (S)	1024	S0 – S777	5120 – 6143	Coil
For Word Data Types ..... Convert PLC Addr. to Dec. +				**Data Type**
Timer Current Values (V)	256	V0 – V177	0 – 255	Input Register
Counter Current Values (V)	256	V1000 – V1177	512 – 767	Input Register
V-Memory, user data (V)	14.6K	V400 – V777 V1400 – V7377 V10000 – V35777	1024 – 2047	Holding Register
V-Memory, system (V)	256 1024	V7400 – V7777 V36000 – V37777	3480 – 4095 15360 – 16383	Holding Register

The following examples show how to generate the Modbus address and data type for hosts which require this format.

Example 1: V2100

Find the Modbus address for User V location V2100.

1. Find V memory in the table.
2. Convert V2100 into decimal (1089).
3. Use the Modbus data type from the table.

PLC Address (Dec.) + Data Type

V2100 = 1088 decimal

1088 + Hold Reg. = **Holding Reg. 1089**

Timer Current Values (V)	128	V0 - V177	0 - 127	Input Register
Counter Current Values (V)	128	V1000 - V1177	512 - 639	Input Register
V Memory, user data (V)	1024	V2000 - -V3777	1024 - 2047	Holding Register

Example 2: Y20

Find the Modbus address for output Y20.

1. Find Y outputs in the table.
2. Convert Y20 into decimal (16).
3. Add the starting address for the range (2049).
4. Use the Modbus data type from the table.

PLC Addr. (Dec) + Start Addr. + Data Type

Y20 = 16 decimal

16 + 2049 + Coil = **Coil 2065**

Outputs (Y)	320	Y0 – Y477	2049 – 2367	Coil
Control Relays (CR)	256	C0 – C377	3072 - 3551	Coil

Example 3: T10 Current Value

Find the Modbus address to obtain the current value from Timer T10.

1. Find Timer Current Values in the table.
2. Convert T10 into decimal (8).
3. Use the Modbus data type from the table.

PLC Address (Dec.) + Data Type

T10 = 8 decimal

8 + Input Reg. = **Input Reg. 9**

Timer Current Values (V)	128	V0 – V177	0 – 128	Input Register
Counter Current Values (V)	128	V1000 – V1177	512 – 639	Input Register

Example 4: C54

Find the Modbus address for Control Relay C54.

1. Find Control Relays in the table.
2. Convert C54 into decimal (44).
3. Add the starting address for the range (3073).
4. Use the Modbus data type from the table.

PLC Addr. (Dec) + Start Addr. +Data Type

C54 = 44 decimal

44 + 3073 + Coil = **Coil 3117**

Outputs (Y)	320	Y0 – Y477	2048 - 2367	Coil
Control Relays (C)	256	C0 – C377	3073 – 3551	Coil

If Your Modbus Host Software Requires an Address ONLY

Some host software does not allow you to specify the Modbus data type and address. Instead, you specify an address only. This method requires another step to determine the address, but it is not difficult. Basically, Modbus separates the data types by address ranges as well. So this means an address alone can actually describe the type of data and location. This is often referred to as "adding the offset." One important thing to remember here is that two different addressing modes may be available in your host software package. These are:

- 484 Mode
- 584/984 Mode

We recommend that you use the 584/984 addressing mode if your host software allows you to choose. This is because the 584/984 mode allows access to a higher number of memory locations within each data type. If your software only supports 484 mode, then there may be some PLC memory locations that will be unavailable. The actual equation used to calculate the address depends on the type of PLC data you are using. The PLC memory types are split into two categories for this purpose.

- Discrete – X, GX, SP, Y, R, S, T, CT (contacts), C (contacts)
- Word – V, Timer current value, Counter current value

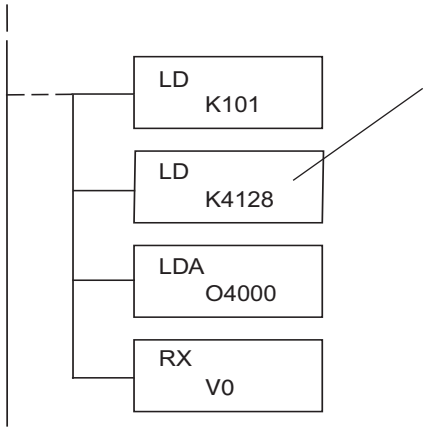
In either case, you basically convert the PLC octal address to decimal and add the appropriate Modbus addresses (as required). The table below shows the exact equation used for each group of data.

Discrete Data Types				
D2-260/D2-262 Memory Type	**PLC Range (Octal)**	**Address Range (484 Mode)**	**Address Range (584/984 Mode)**	**Modbus Data Type**
Global Inputs (GX)	GX0 – GX1746	1001 – 1999	10001 – 10999	Input
	GX1747 – GX3777	---	11000 – 12048	Input
Inputs (X)	X0 – X1777	---	12049 – 13072	Input
Special Relays (SP)	SP0 – SP777	---	13073 – 13584	Input
Global Outputs (GY)	GY0 – GY3777	1 – 2048	1 – 2048	Output
Outputs (Y)	Y0 – Y1777	2049 – 3072	2049 – 3072	Output
Control Relays (C)	C0 – C3777	3073 – 5120	3073 – 5120	Output
Timer Contacts (T)	T0 – T377	6145 – 6400	6145 – 6400	Output
Counter Contacts (CT)	CT0 – CT377	6401 – 6656	6401 – 6656	Output
Stage Status Bits (S)	S0 – S1777	5121 – 6144	5121 – 6144	Output

Word Data Types			
Registers	**PLC Range (Octal)**	**Input*/Holding (484 Mode)**	**Input*/Holding (585/984 Mode)**
V-Memory (Timers)	V0 – V377	3001/4001	30001/40001
V-Memory (Counters)	V1000 – V1177	3513/4513	30513/40513
V-Memory (Data Words)	V1200 – V1377	3641/4641	30641/40641
	V1400 – V1746	3769/4769	30769/40769
	V1747 – V1777	---	31000/41000
	V2000 – V7377	---	41025
	V10000 – V17777	---	44097

***Modbus: Function 04**

The D2-250-1, D2-260 and D2-262 support **function 04** read input register (**Address 30001**). To use function 04, put the number '4' into the most significant position (4xxx) when defining the number of bytes to read. Four digits must be entered for the instruction to work properly with this mode.



The maximum constant possible is 4128. This is due to the 128 maximum number of Bytes that the RX/WX instruction can allow. The value of 4 in the most significant position of the word will cause the RX to use function 04 (30001 range).

Later in this chapter, a step-by-step procedure will provide the information necessary to set up the ladder program to receive data from a network slave.

Refer to your PLC user manual for the correct memory size of your PLC. Some of the addresses shown above might not pertain to your particular CPU.

For an automated Modbus/Koyo address conversion utility, search and download the file **modbus_conversion.xls** from the **www.automationdirect.com** website.

Example 1: V2100 584/984 Mode

Find the Modbus address for User V location V2100.

1. Find V memory in the table.
2. Convert V2100 into decimal (1088).
3. Add the Modbus starting address for the mode (40001).

PLC Address (Dec.) + Mode Address

V2100 = 1088 decimal

1088 + 40001 = 41089

For Word Data Types...	PLC Address (Dec.)		+	Appropriate Mode Address		
Timer Current Value (V)	128	V0 - V177	0 - 127	3001	30001	Input Register
Counter Current Value (V)	128	V1000 - V1177	512 - 639	3001	30001	Input Register
V Memory, User Data (V)	1024	V2000 - V3777	1024 - 2047	4001	40001	Hold Register

Example 2: Y20 584/984 Mode

Find the Modbus address for output Y20.

1. Find Y outputs in the table.
2. Convert Y20 into decimal (16).
3. Add the starting address for the range (2048).
4. Add the Modbus address for the mode (1).

PLC Addr. (Dec.) + Start Address + Mode

Y20 = 16 decimal

16 + 2048 + 1 = **2065**

Outputs (Y)	320	Y0 - Y477	2048 - 2367	1	1	Coil
Control Relays (CR)	256	C0 - C377	3072 - 3551	1	1	Coil
Timer Contacts (T)	128	T0 - T177	6144 - 6271	1	1	Coil

Example 3: T10 Current Value 484 Mode

Find the Modbus address to obtain the current value from Timer T10.

1. Find Timer Current Values in the table.
2. Convert T10 into decimal (8).
3. Add the Modbus starting address for the mode (3001).

PLC Address (Dec.) + Mode Address

TA10 = 8 decimal

8 + 3001 = **3009**

For Word Data Types...	PLC Address (Dec.)		+	Appropriate Mode Address		
Timer Current Value (V)	128	V0 - V177	0 - 127	3001	30001	Input Register
Counter Current Value (V)	128	V1000 - V1177	512 - 639	3001	30001	Input Register
V Memory, User Data (V)	1024	V2000 - V3777	1024 - 2047	4001	40001	Hold Register

Example 4: C54 584/984 Mode

Find the Modbus address for Control Relay C54.

1. Find Control Relays in the table.
2. Convert C54 into decimal (44).
3. Add the starting address for the range (3072).4. Add the Modbus address for the mode (1).

PLC Addr. (Dec.) + Start Address + Mode

C54 = 44 decimal

44 + 3072 + 1 = **3117**

Outputs (Y)	320	Y0 - Y477	2048 - 2367	1	1	Coil
Control Relays (CR)	256	C0 - C377	3072 - 3551	1	1	Coil
Timer Contacts (T)	128	T0 - T177	6144 - 6271	1	1	Coil

Determining the DirectNET Address

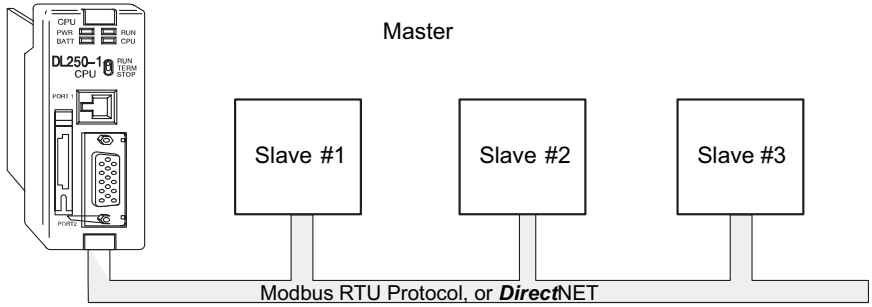
☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

Addressing the memory types for *Direct*NET slaves is very easy. Use the ordinary native address of the slave device itself. To access a slave PLC's memory address V2000 via *Direct*NET, for example, the network master will request V2000 from the slave.

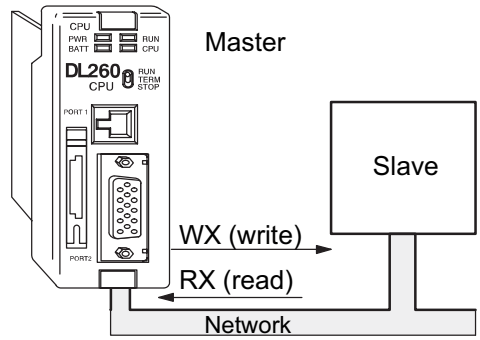
Network Master Operation

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

This section describes how the D2-250–1, D2-260 and D2-262 can communicate on a Modbus or *Direct*NET network as a master. For Modbus networks, it uses the Modbus RTU protocol, which must be interpreted by all the slaves on the network. Both Modbus and *Direct*NET are single master/multiple slave networks. The master is the only member of the network that can initiate requests on the network. This section teaches you how to design the required ladder logic for network master operation.



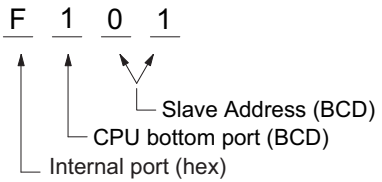
When using the D2-250-1, D2-260 or D2-262 CPU as the master station, you use simple RLL instructions to initiate the requests. The WX instruction initiates network write operations, and the RX instruction initiates network read operations. Before executing either the WX or RX commands, we will need to load data related to the read or write operation onto the CPU's accumulator stack. When the WX or RX instruction executes, it uses the information on the stack combined with data in the instruction box to completely define the task, which goes to the port.



The following step-by-step procedure will provide the information necessary to set up your ladder program to receive data from a network slave.

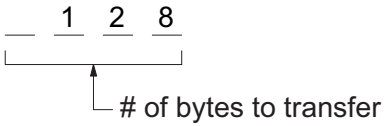
Step 1: Identify Master Port # and Slave

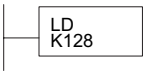
The first Load (LD) instruction identifies the communications port number on the network master (D2-250-1/D2-260/D2-262) and the address of the slave station. This instruction can address up to 99 Modbus slaves, or 90 *Direct*NET slaves. The format of the word is shown to the right. The "F1" in the upper byte indicates the use of the bottom port of the D2-250-1, D2-260 and D2-262 PLC, port number 2. The lower byte contains the slave address number in BCD (01 to 99).



Step 2: Load Number of Bytes to Transfer

The second Load (LD) instruction determines the number of bytes which will be transferred between the master and slave in the subsequent WX or RX instruction. The value to be loaded is in BCD format (decimal), from 1 to 128 bytes.





The number of bytes specified also depends on the type of data you want to obtain. For example, the DL205 Input points can be accessed by V-memory locations or as X input locations. However, if you only want X0 – X27, you'll have to use the X input data type because the V-memory locations can only be accessed in 2-byte increments. The following table shows the byte ranges for the various types of *Direct*LOGIC™ products.

DL205/405 Memory	Bits per unit	Bytes
V-memory T / C current value	16 16	2 2
Inputs (X, SP)	8	1
Outputs (Y, C, Stage, T/C bits)	8	1
Scratch Pad Memory	8	1
Diagnostic Status	8	1

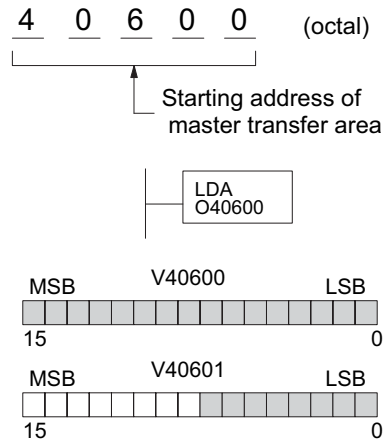
DL305 Memory	Bits per unit	Bytes
Data registers T / C accumulator	8 16	1 2
I/O, internal relays, shift register bits, T/C bits, stage bits	1	1
Scratch Pad Memory	8	2
Diagnostic Status (5 word R/W)	16	10

Step 3: Specify Master Memory Area

The third instruction in the RX or WX sequence is a Load Address (LDA) instruction. Its purpose is to load the starting address of the memory area to be transferred. Entered as an octal number, the LDA instruction converts it to hex and places the result in the accumulator.

For a WX instruction, the D2-250-1, D2-260 or D2-262 CPU sends the number of bytes previously specified from its memory area beginning at the LDA address specified.

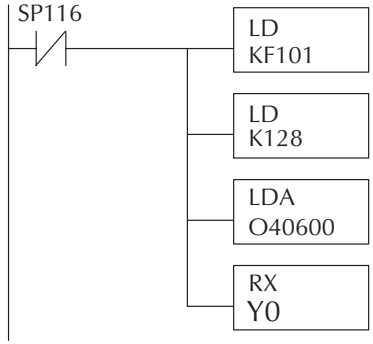
For an RX instruction, the D2-250-1, D2-260 or D2-262 CPU reads the number of bytes previously specified from the slave, placing the received data into its memory area beginning at the LDA address specified.



NOTE: *Since V-memory words are always 16 bits, you may not always use the whole word. For example, if you only specify 3 bytes and you are reading Y outputs from the slave, you will only get 24 bits of data. In this case, only the 8 least significant bits of the last word location will be modified. The remaining 8 bits are not affected.*

Step 4: Specify Slave Memory Area

The last instruction in our sequence is the WX or RX instruction itself. Use WX to write to the slave, and RX to read from the slave. All four of our instructions are shown to the right. In the last instruction, you must specify the starting address and a valid data type for the slave.

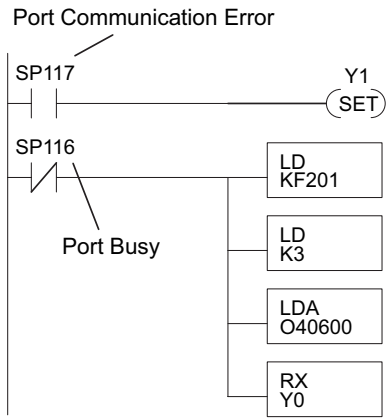


- DirectNET slaves – specify the same address in the WX and RX instruction as the slave's native I/O address.
- Modbus DL405 or DL205 slaves – specify the same address in the WX and RX instruction as the slave's native I/O address.
- Modbus 305 slaves – use the following table to convert DL305 addresses to Modbus addresses.

DL305 Series CPU Memory Type–to–Modbus Cross Reference					
PLC Memory Type	**PLC Base Address**	**Modbus Base Address**	**PLC Memory Type**	**PLC Base Address**	**Modbus Base Address**
TMR/CNT Current Values	R600	V0	TMR/CNT Status Bits	CT600	GY600
I/O Points	IO 000	GY0	Control Relays	CR160	GY160
Data Registers	R401,R400	V100	Shift Registers	SR400	GY400
Stage Status Bits (D3-330P only)	S0	GY200			

Communications from a Ladder Program

Typically, network communications will last longer than one scan. The program must wait for the communications to finish before starting the next transaction.



Port 2, which can be a master, has two Special Relay contacts associated with it. One indicates "Port busy"(SP116), and the other indicates "Port Communication Error"(SP117). The example shows the use of these contacts for a network master that only reads a device (RX). The "Port Busy" bit is on while the PLC communicates with the slave. When the bit is off, the program can initiate the next network request.

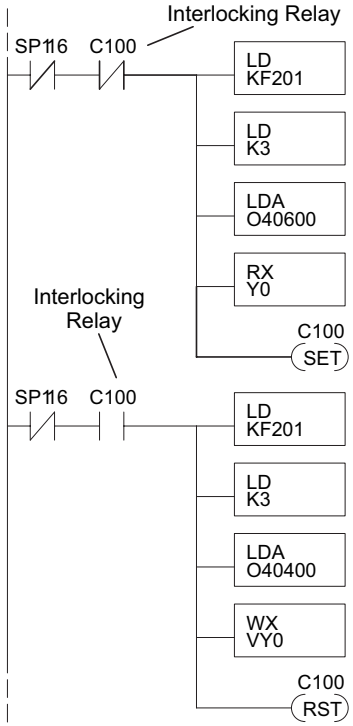
The "Port Communication Error" bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an RX or WX instruction is executed.

Multiple Read and Write Interlocks

If you are using multiple reads and writes in the RLL program, you have to interlock the routines to make sure all the routines are executed. If you don't use the interlocks, then the CPU will only execute the first routine. This is because each port can only handle one transaction at a time.

In the example to the right, after the RX instruction is executed, C100 is set. When the port has finished the communication task, the second routine is executed and C100 is reset.

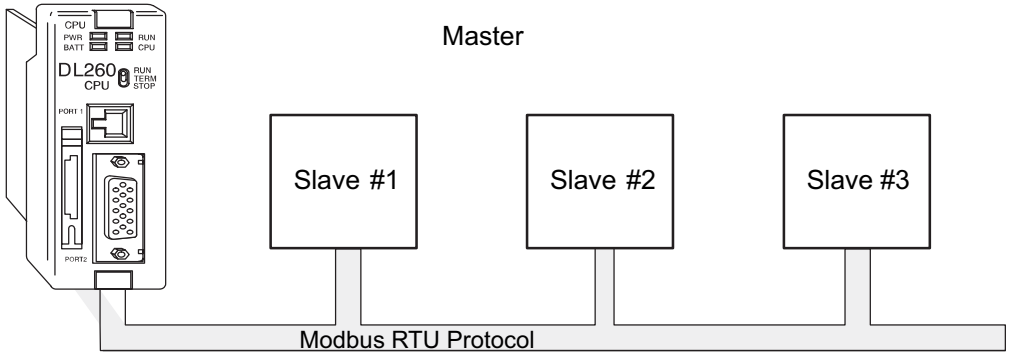
If you're using RLL*PLUS* Stage Programming, you can put each routine in a separate program stage to ensure proper execution and switch from stage to stage allowing only one of them to be active at a time.



Network Modbus RTU Master Operation (D2-260 and D2-262 only)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

This section describes how the D2-260 and D2-262 CPUs can communicate on a Modbus RTU network as a master using the MRX and MWX read/write instructions. These instructions allow you to enter native Modbus addressing in your ladder logic program with no need to perform octal-to-decimal conversions. Modbus is a single-master, multiple-slave network. The master is the only member of the network that can initiate requests on the network. This section teaches you how to design the required ladder logic for network master operation.



Modbus Function Codes Supported

The Modbus function code determines whether the access is a read or a write, and whether to access a single data point or a group of them. The D2-260 and D2-262 CPUs supports the Modbus function codes described below.

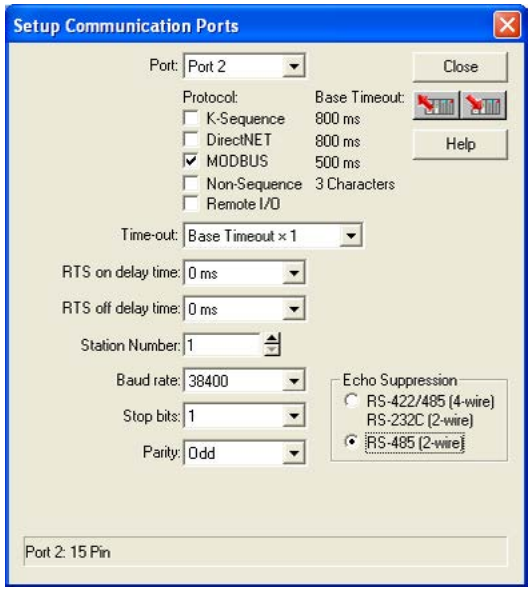
Modbus Function Code	Function	DL205 Data Types Available
01	Read a group of coils	Y, C, T, CT
02	Read a group of inputs	X, SP
05	Set / Reset a single coil (slave only)	Y, C, T, CT
15	Set / Reset a group of coils	Y, C, T, CT
03, 04	Read a value from one or more registers	V
06	Write a value into a single register (slave only)	V
07	Read Exception Status	V
08	Diagnostics	V
16	Write a value into a group of registers	V

Modbus Port Configuration

In *Direct*SOFT, choose the PLC menu, then Setup, then "Secondary Comm Port."

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

- **Port:** From the port number list box at the top, choose "Port 2."
- **Protocol**: Click the check box to the left of "MODBUS" (use AUX 56 on the HPP, and select "MBUS"), and then you'll see the dialog box below.
- **Timeout:** Amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS On Delay Time:** The amount of time between raising the RTS line and sending the data.
- **RTS Off Delay Time:** The amount of time between resetting the RTS line after sending the data.
- **Station Number:** For making the CPU port a Modbus master, choose "1." The possible range for Modbus slave numbers is from 1 to 247. Each slave must have a unique number. At powerup, the port is automatically a slave, unless and until the D2-260 or D2-262 CPUs execute ladder logic with MWX/MRX network instructions, which use the port as a communications master. Thereafter, the port reverts back to slave mode until ladder logic uses the port again.



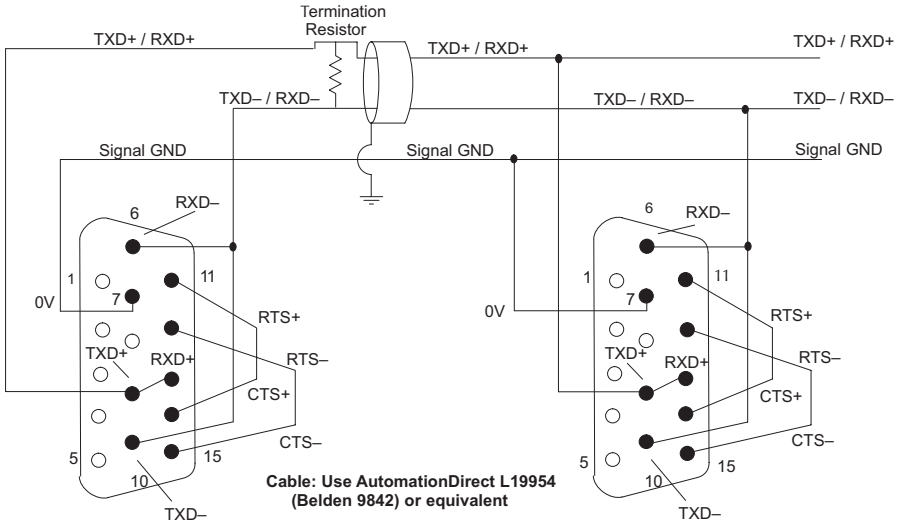
- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value. Refer to the appropriate product manual for details.
- **Stop Bits:** Choose 1 or 2 stop bits for use in the protocol.
- **Parity:** Choose none, even, or odd parity for error checking.
- **Echo Suppression:** Select the appropriate radio button based on the wiring configuration used on port 2.

Then click the button indicated to send the Port configuration to the CPU, and click Close.

RS–485 Network (Modbus Only)

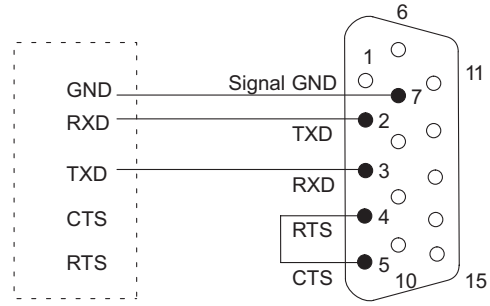
☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

RS–485 signals are for longer distances (1000 meters maximum), and for multi-drop networks. Use termination resistors at both ends of RS–485 network wiring, matching the impedance rating of the cable (between 100 and 500 ohms).



RS–232 Network

Normally, the RS–232 signals are used for shorter distances (15 meters maximum), for communications between two devices.

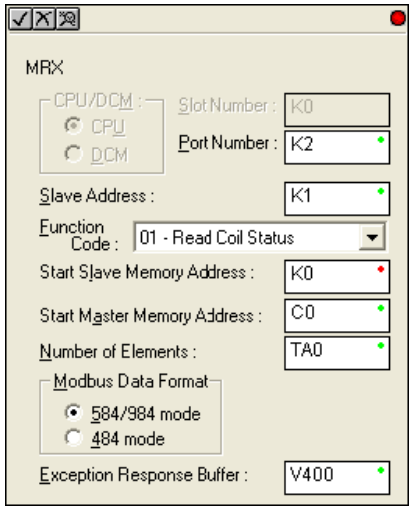


Port 2 Pin Descriptions (D2-260/D2-262 only)		
1	5V	5 VDC
2	TXD2	Transmit Data (RS-232)
3	RXD2	Receive Data (RS-232)
4	RTS2	Ready to Send (RS–232)
5	CTS2	Clear to Send (RS–232)
6	RXD2–	Receive Data – (RS–422/RS-485)
7	0V	Logic Ground
8	0V	Logic Ground
9	TXD2+	Transmit Data + (RS–422/RS–485)
10	TXD2 –	Transmit Data – (RS–422/RS–485)
11	RTS2 +	Request to Send + (RS–422/RS–485)
12	RTS2 –	Request to Send – (RS–422/RS–485)
13	RXD2 +	Receive Data + (RS–422/RS–485)
14	CTS2 +	Clear to Send + (RS422/RS–485)
15	CTS2 –	Clear to Send – (RS–422/RS–485)

Modbus Read from Network (MRX)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

The Modbus Read from Network (MRX) instruction is used by the D2-260 or D2-262 network master to read a block of data from a connected slave device and to write the data into memory addresses within the master. The instruction allows the user to specify the Modbus Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, Modbus data format and the Exception Response Buffer.



- **Port Number:** must be Port 2 (K2)
- **Slave Address:** specify a slave station address (1–247)
- **Function Code**: the MRX instruction supports the following Modbus function codes:
  - 01 – Read a group of coils
  - 02 – Read a group of inputs
  - 03 – Read holding registers
  - 04 – Read input registers
  - 07 – Read Exception status
- **Start Slave Memory Address**: specifies the starting slave memory address of the data to be read. See the table on the following page.
- **Start Master Memory Address**: specifies the starting memory address in the master where the data will be placed. See the table on the following page.
- **Number of Elements**: specifies how many coils, discrete inputs, holding registers or input registers will be read. See the table on the following page.
- **Modbus Data Format**: specifies Modbus 584/984 or 484 data format to be used.
- **Exception Response Buffer**: specifies the master memory address where the Exception Response will be placed. See the table on the following page.

MRX Slave Memory Address

MRX Slave Address Ranges		
Function Code	**Modbus Data Format**	**Slave Address Range(s)**
01 – Read Coil	484 Mode	1–999
01 – Read Coil	584/984 Mode	1–65535
02 – Read Input Status	484 Mode	1001–1999
02 – Read Input Status	584/984 Mode	10001–19999 (5 digit) or 100001–165535 (6 digit)
03 – Read Holding Register	484 Mode	4001–4999
03 – Read Holding Register	584/984	40001–49999 (5 digit) or 4000001–465535 (6 digit)
04 – Read Input Register	484 Mode	3001–3999
04 – Read Input Register	584/984 Mode	30001–39999 (5 digit) or 3000001–365535 (6 digit)
07 – Read Exception Status	484 and 584/984 Mode	N/A

MRX Master Memory Addresses

MRX Master Memory Address Ranges		
Operand Data Type		**D2-260/D2-262 Range**
Inputs	X	0–1777
Outputs	Y	0–1777
Control Relays	C	0–3777
Stage Bits	S	0–1777
Timer Bits	T	0–377
Counter Bits	CT	0–377
Special Relays	SP	0–777
V–memory	V	All
Global Inputs	GX	0–3777
Global Outputs	GY	0–3777

MRX Number of Elements

Number of Elements		
Operand Data Type		**D2-260/D2-262 Range**
V–memory	V	All (see page 3-56)
Constant	K	Bits:1–2000 Registers: 1-125

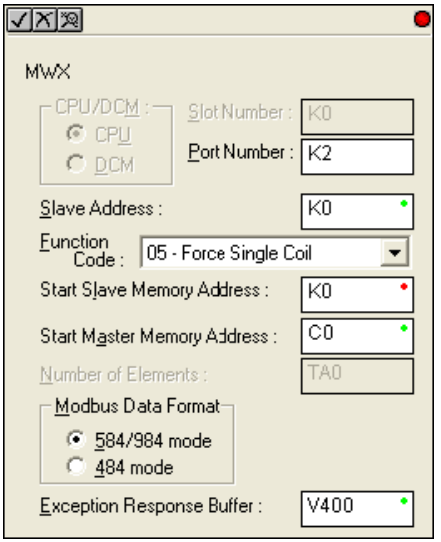
MRX Exception Response Buffer

Exception Response Buffer		
Operand Data Type		**D2-260/D2-262 Range**
V–memory	V	All (see page 3-56)

Modbus Write to Network (MWX)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

The Modbus Write to Network (MWX) instruction is used by the D2-260 or D2-262 network master to write a block of data to Modbus memory addresses within a slave device on the network. The instruction allows the user to specify the Modbus Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, Modbus data format and the Exception Response Buffer.



- **Port Number**: must be Port 2 (K2).
- **Slave Address**: specify a slave station address (0–247).
- **Function Code**: the MWX instruction supports the following Modbus function codes:
  - 05 – Force Single coil
  - 06 – Preset Single Register
  - 08 – Diagnostics
  - 15 – Force Multiple Coils
  - 16 – Preset Multiple Registers
- **Start Slave Memory Address**: specifies the starting slave memory address where the data will be written.
- **Start Master Memory Address:** specifies the starting address of the data in the master that is to written to the slave.
- **Number of Elements**: specifies how many consecutive coils or registers will be written to. This field is only active when either function code 15 or 16 is selected.
- **Modbus Data Format**: specifies Modbus 584/984 or 484 data format to be used.
- **Exception Response Buffer**: specifies the master memory address where the Exception Response will be placed.

MWX Slave Memory Address

MWX Slave Address Ranges		
Function Code	**Modbus Data Format**	**Slave Address Range(s)**
05 – Force Single Coil	484 Mode	1–999
05 – Force Single Coil	584/984 Mode	1–65535
06 – Preset Single Register	484 Mode	4001–4999
06 – Preset Single Register	584/984 Mode	40001–49999 (5 digit) or 400001–465535 (6 digit)
15 – Force Multiple Coils	484	1–999
15 – Force Multiple Coils	584/984 Mode	1–65535
16 – Preset Multiple Registers	484 Mode	4001–4999
16 – Preset Multiple Registers	584/984 Mode	40001–49999 (5 digit) or 4000001–465535 (6 digit)

MWX Master Memory Addresses

MRX Master Memory Address Ranges		
Operand Data Type		**D2-260/D2-262 Range**
Inputs	X	0–1777
Outputs	Y	0–1777
Control Relays	C	0–3777
Stage Bits	S	0–1777
Timer Bits	T	0–377
Counter Bits	CT	0–377
Special Relays	SP	0–777
V–memory	V	All (see page 3-56)
Global Inputs	GX	0–3777
Global Outputs	GY	0–3777

MWX Number of Elements

Number of Elements		
Operand Data Type		**D2-260/D2-262 Range**
V–memory	V	All (see page 3-56)
Constant	K	Bits:1–2000 Registers: 1-125

MWX Exception Response Buffer

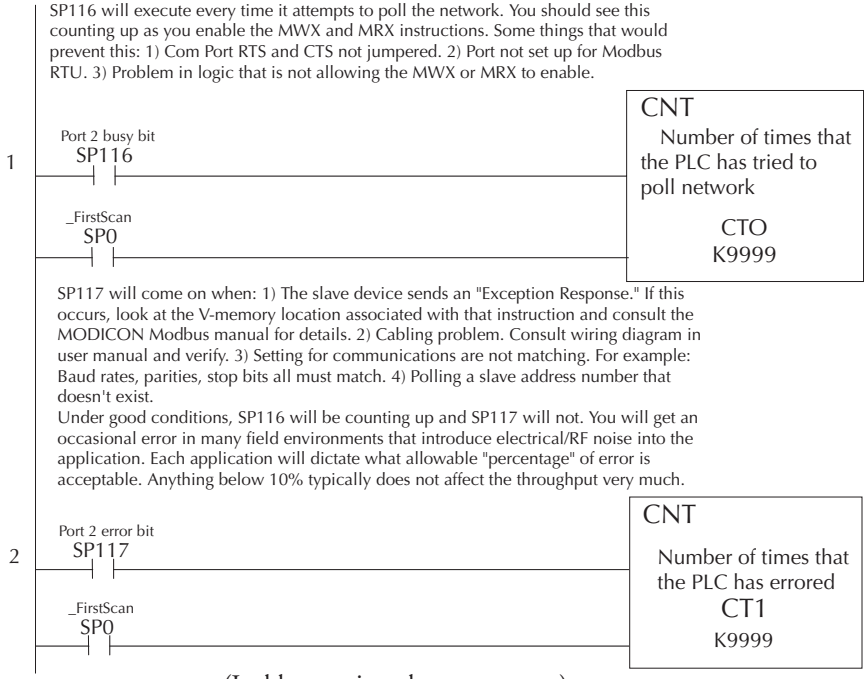
Exception Response Buffer		
Operand Data Type		**D2-260/D2-262 Range**
V–memory	V	All (see page 3-56)

MRX/MWX Example in DirectSOFT

D2-260 and D2-262 port 2 has two Special Relay contacts associated with it (see Appendix D for comm port special relays). One indicates "Port busy"(SP116), and the other indicates "Port Communication Error"(SP117). The "Port Busy" bit is on while the PLC communicates with the slave. When the bit is off, the program can initiate the next network request. The "Port Communication Error" bit turns on when the PLC has detected an error and use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an MRX or MWX instruction is executed. Typically, network communications will last longer than one CPU scan. The program must wait for the communications to finish before starting the next transaction.

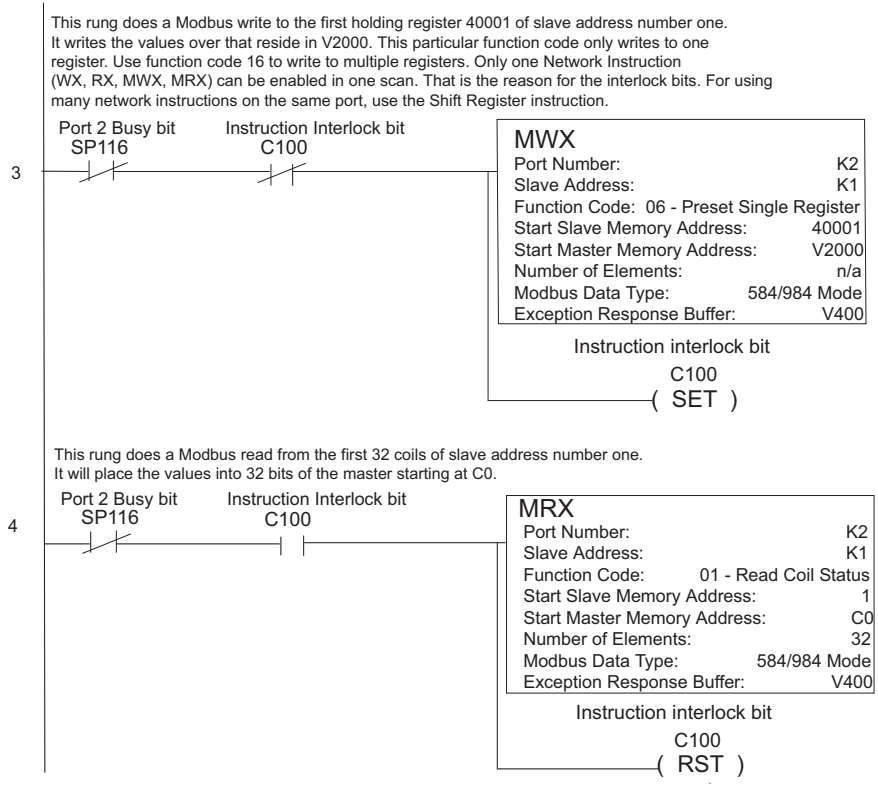
Multiple Read and Write Interlocks

If you are using multiple reads and writes in the RLL program, you need to interlock the routines to make sure all the routines are executed. If you don't use the interlocks, then the CPU will only execute the first routine. This is because each port can only handle one transaction at a time. In the example, rungs 3 and 4 show that C100 will get set after the RX instruction has been executed. When the port has finished the communication task, the second routine is executed and C100 is reset. If you're using RLL*PLUS* Stage Programming, you can put each routine in a separate program stage to ensure proper execution and switch from stage to stage allowing only one of them to be active at a time.



(Ladder continued on next page.)

Continued from previous page.



Non–Sequence Protocol (ASCII In/Out and PRINT)

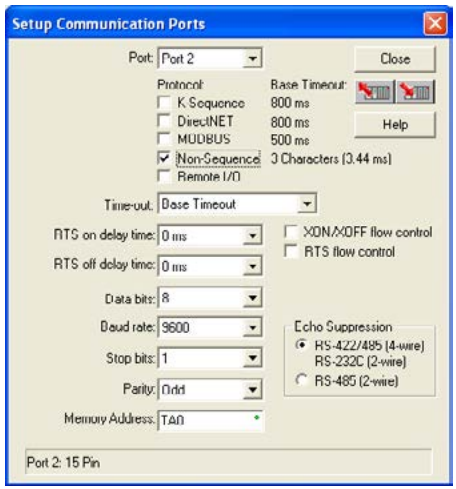
Configure the D2-260 and D2-262 CPUs for Non-Sequence

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

Configuring port 2 on the D2-260 and D2-262 CPUs for Non–Sequence allows the CPUs to use port 2 to either read or write raw ASCII strings using the ASCII instructions. See the ASCII In/Out instructions and the PRINT instruction in chapter 5.

In *Direct*SOFT, choose the PLC menu, then "Setup Secondary Comm Port."

- **Port**: From the port number list box at the top, choose "Port 2."
- **Protocol**: Click the check box to the left of "Non–Sequence."
- **Timeout**: Amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS On Delay Time**: The amount of time between raising the RTS line and sending the data.
- **RTS Off Delay Time**: The amount of time between resetting the RTS line after sending the data.
- **Data Bits**: Select either 7–bits or 8–bits to match the number of data bits specified for the connected devices.
- **Baud Rate**: The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value. Refer to the appropriate product manual for details.



- **Stop Bits**: Choose 1 or 2 stop bits to match the number of stop bits specified for the connected devices.
- **Parity**: Choose none, even, or odd parity for error checking. Be sure to match the parity specified for the connected devices.
- **Memory Address:** Starting V-memory address for ASCII In data storage. This location is the start of protocol memory buffer. It should not be used for other purposes.
- **Buffer size** = 2 + (Max receiving data size / 2), or to allocate the maximum allowable space, buffer size = 66 Words (for example V2000-V2102).
- **XON/XOFF Flow Control:** When this function is enabled, the PLC will send data (PRINT command) until it receives a XOFF (0x13) Pause transmission command. It will continue to wait until it then sees a XON (0x11) Resume transmission command. This selection is only available when the "Non-Sequence(ASCII)" option has been selected and only functions when the PLC is sending data (not receiving with AIN command).

- **RTS Flow Control:** When this function is enabled, the PLC will assert the RTS signal(s) of the port and wait to see the CTS signal(s) go true before sending data (PRINT command). This selection is only available when the "Non-Sequence(ASCII)" option has been selected and only functions when the PLC is sending data (not receiving with AIN command).
- **Echo Suppression**: Select the appropriate radio button based on the wiring configuration used on port 2.

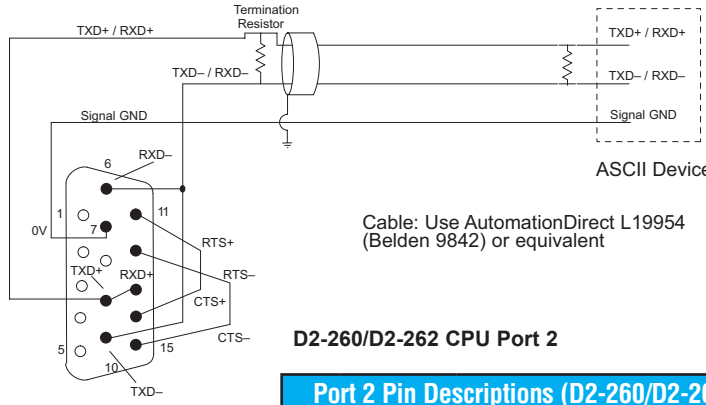
Then click the button indicated to send the Port configuration to the CPU, and click Close.

RS–485 Network

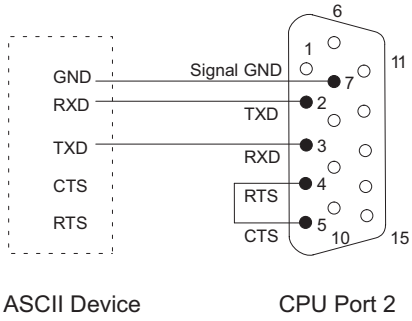
RS–485 signals are for long distances (1000 meters maximum). Use termination resistors at both ends of RS–485 network wiring, matching the impedance rating of the cable (between 100 and 500 ohms).

RS–232 Network

RS–232 signals are used for shorter distances (15 meters maximum) and limited to communications between two devices.



Cable: Use AutomationDirect L19954 (Belden 9842) or equivalent



D2-260/D2-262 CPU Port 2

Port 2 Pin Descriptions (D2-260/D2-262 only)		
1	5V	5 VDC
2	TXD2	Transmit Data (RS-232)
3	RXD2	Receive Data (RS-232)
4	RTS2	Ready to Send (RS–232)
5	CTS2	Clear to Send (RS–232)
6	RXD2–	Receive Data – (RS–422/RS-485)
7	0V	Logic Ground
8	0V	Logic Ground
9	TXD2+	Transmit Data + (RS–422/RS–485)
10	TXD2 –	Transmit Data – (RS–422/RS–485)
11	RTS2 +	Request to Send + (RS–422/RS–485)
12	RTS2 –	Request to Send – (RS–422/RS–485)
13	RXD2 +	Receive Data + (RS–422/RS–485)
14	CTS2 +	Clear to Send + (RS-422/RS–485)
15	CTS2 –	Clear to Send – (RS–422/RS–485)

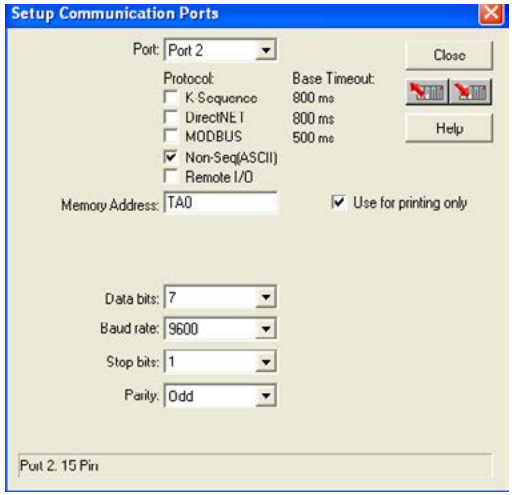
Configure the D2-250-1 Port 2 for Non-Sequence

Configuring port 2 on the D2-250-1 for Non–Sequence enables the CPU to use the PRINT instruction to print embedded text or text/data variable message from port 2. **See the PRINT instruction in chapter 5.**

☐ 230
☐ 240
☑ 250-1
☐ 260

In *Direct*SOFT, choose the PLC menu, then "Setup Secondary Comm Port."

- **Port:** From the port number list box at the top, choose "Port 2."
- **Protocol:** Click the check box to the left of "Non–Sequence."
- **Memory Address:** Choose a V-memory address to use as the starting location for the port set-up parameters listed below. This location is the start of protocol memory buffer. It should not be used for other purposes. Buffer size = 2 + (Max receiving data size) / 2 or to allocate the maximum allowable space buffer size = 66 Words (for example V2000-V2102).
- **Use For Printing Only:** Check the box to enable the port settings described below. Match the settings to the connected device.



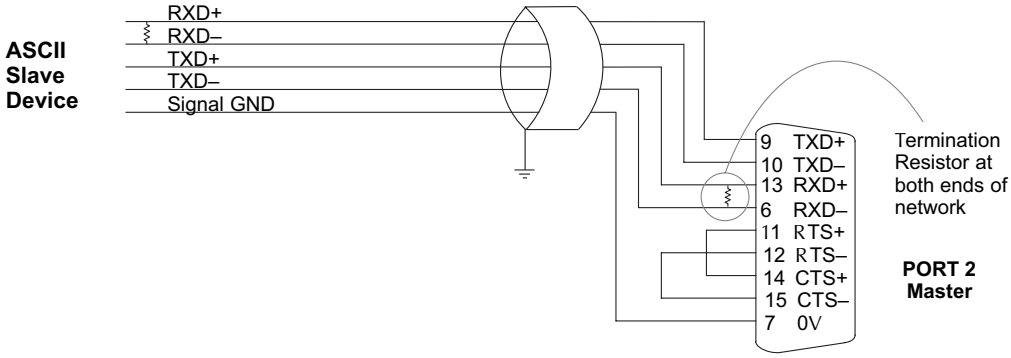
- **Data Bits:** Select either 7–bits or 8–bits to match the number of data bits specified for the connected device.
- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value. Refer to the appropriate product manual for details.
- **Stop Bits:** Choose 1 or 2 stop bits to match the number of stop bits specified for the connected device.
- **Parity:** Choose none, even, or odd parity for error checking. Be sure to match the parity specified for the connected device.

Then click the button indicated to send the Port configuration to the CPU, and click Close.

RS–422 Network

RS–422 signals are for long distances (1000 meters max.). Use termination resistors at both ends of RS–422 network wiring, matching the impedance rating of the cable (between 100 and 500 ohms).

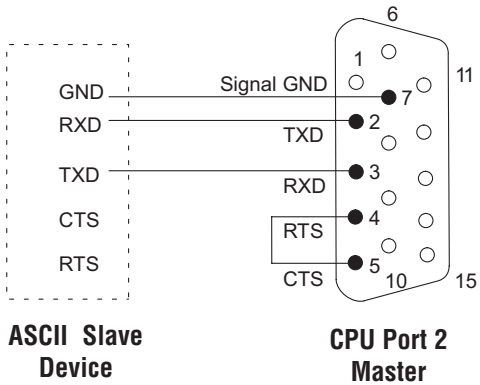
NOTE: *For RS–422 cabling, we recommend AutomationDirect L19853 (Belden 8103) or equivalent.*



RS–232 Network

RS–232 signals are used for shorter distances (15 meters maximum) and limited to communications between two devices.

NOTE: *For RS–232 cabling, we recommend AutomationDirect L19772 (Belden 8102) or equivalent.*



Port 2 Pin Descriptions (D2-250-1)		
1	5V	5 VDC
2	TXD2	Transmit Data (RS-232)
3	RXD2	Receive Data (RS-232)
4	RTS2	Ready to Send (RS–232)
5	CTS2	Clear to Send (RS–232)
6	RXD2–	Receive Data – (RS–422)
7	0V	Logic Ground
8	0V	Logic Ground
9	TXD2+	Transmit Data + (RS–422)
10	TXD2 –	Transmit Data – (RS–422)
11	RTS2 +	Request to Send + (RS–422)
12	RTS2 –	Request to Send – (RS–422)
13	RXD2 +	Receive Data + (RS–422)
14	CTS2 +	Clear to Send + (RS422)
15	CTS2 –	Clear to Send – (RS–422)

RLL and Intelligent Box Instructions

In This Chapter...

Introduction

The DL205 CPUs offer a wide variety of instructions to perform many different types of operations. Several instructions are not available in all of the CPUs. This chapter shows you how to use these individual instructions. There are two ways to quickly find the instruction you need:

- If you know the instruction category (Boolean, Comparative Boolean, etc), use the header at the top of the page to find the pages that discuss the instructions in that category.
- If you know the individual instruction name, use the following table to find the page that discusses the instruction.

Instruction		Page
FAULT	Fault	5-197
FDGT	Find Greater Than	5-152
FILL	Fill	5–150
FIND	Find	5–151
FINDB	Find Block	5–173
FOR	For/Next	5–180
GOTO	Goto/Label	5–179
GRAY	Gray Code	5–141
GTS	Goto Subroutine	5–182
HTA	Hex to ASCII	5–138
INC	Increment	5–100
INCB	Increment Binary	5–107
INT	Interrupt	5–187
INV	Invert	5–132
IRT	Interrupt Return	5–188
IRTC	Interrupt Return Conditional	5–188
ISG	Initial Stage	7–24
JMP	Jump	7–24
LBL	Label	5–179
LD	Load	5–58
LDI	Load Immediate	5–39
LDIF	Load Immediate Formatted	5–40
LDA	Load Address	5-61
LDD	Load Double	5–59
LDF	Load Formatted	5–60
LDR	Load Real Number	5–64
LDX	Load Indexed	5–62
LDLBL	Load Label	5–145
LDSX	Load Indexed from Constant	5–63
MDRMD	Masked Drum Event Discrete	6–19
MDRMW	Masked Drum Event Word	6–21
MLR	Master Line Reset	5–185
MLS	Master Line Set	5–185
MOV	Move	5–144
MOVMC	Move Memory Cartridge	5–145
MRX	Read from MODBUS Network	5–205
MWX	Write to MODBUS	5–208
MUL	Multiply	5–94
MULB	Multiply Binary	5–105
MULBS	Multiply Binary top of stack	5–119
MULD	Multiply Double	5–95
MULF	Multiply Formatted	5–111
MULR	Multiply Real	5–96
MULS	Multiply Top of Stack	5–115
NCON	Numeric Constand	5–199
NEXT	Next (For/Next)	5–180

Instruction		Page
NJMP	Not Jump (Stage)	7–24
NOP	No Operation	5-177
NOT	Not	5–19
OR	Or	5–12, 5–31, 5–75
OR OUT	Or Out	5–19
OR OUTI	Or Out Immediate	5–36
OR STR	Or Store	5–16
ORB	Or Bit–of–Word	5–13
ORD	Or Double	5–76
ORE	Or if Equal	5–28
ORF	Or Formatted	5–77
ORI	Or Immediate	5–34
ORMOV	Or Move	5–171
ORN	Or Not	5–12, 5–31
ORNB	Or Not Bit–of–Word	5–13
ORND	Or Negative Differential	5–22
ORNE	Or if Not Equal	5–28
ORNI	Or Not Immediate	5–34
ORPD	Or Positive Differential	5–22
ORS	Or Stack	5–78
OUT	Out	5–17, 5–65
OUTB	Out Bit–of–Word	5–18
OUTD	Out Double	5–66
OUTF	Out Formatted	5–67
OUTI	Out Immediate	5–36
OUTIF	Out Immediate Formatted	5–37
OUTL	Out Least	5–69
OUTM	Out Most	5–69
OUTX	Out Indexed	5–68
PAUSE	Pause	5–26
PD	Positive Differential	5–20
POP	Pop	5–70
PRINT	Print	5–201
PRINTV	ASCII Print from V–Memory	5–227
RADR	Radian Real Conversion	5–136
RD	Read from Intelligent Module	5–191
RFB	Remove from Bottom of Table	5–157
RFT	Remove from Top of Table	5–163
ROTL	Rotate Left	5–126
ROTR	Rotate Right	5–127
RST	Reset	5–24
RSTB	Reset Bit–of–Word	5–25
RSTBIT	Reset Bit	5–148
RSTI	Reset Immediate	5–38
RSTWT	Reset Watch Dog Timer	5–178

Using Boolean Instructions

Do you ever wonder why so many PLC manufacturers always quote the scan time for a 1K boolean program? Simple, most programs utilize many boolean instructions. These are typically very simple instructions designed to join input and output contacts in various series and parallel combinations. Our *Direct*SOFT programming package is a similar program. It uses graphic symbols to develop a program; therefore, you don't necessarily have to know the instruction mnemonics in order to develop your program.

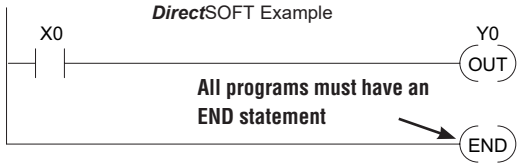
Many of the instructions in this chapter are not program instructions used in *Direct*SOFT, but are implied. In other words, they are not actually keyboard commands but they can be seen in a Mnemonic View of the program once the *Direct*SOFT program has been developed and accepted (compiled). Each instruction listed in this chapter will have a small chart to indicate how the instruction is used with *Direct*SOFT and the HPP.

DS	Implied
HPP	Used

The following paragraphs describe how these instructions are used to build simple ladder programs.

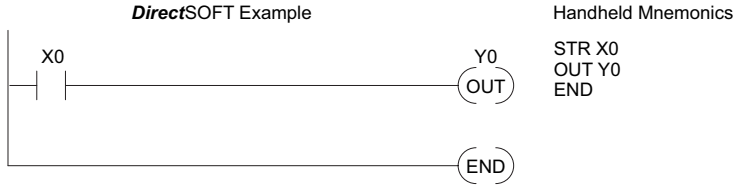
END Statement

All DL205 programs require an END statement as the last instruction. This tells the CPU that this is the end of the program. Normally, any instructions placed after the END statement will not be executed. There are exceptions to this such as interrupt routines, etc. Chapter 5 discusses the instruction set in detail.



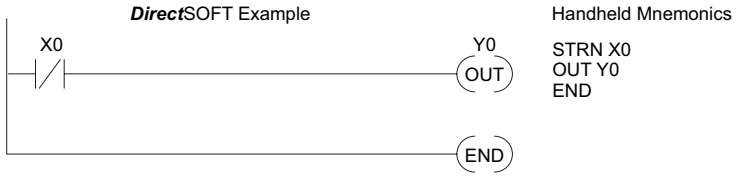
Simple Rungs

You use a contact to start rungs that contain both contacts and coils. The boolean instruction that does this is called a Store or, STR instruction. The output point is represented by the Output or, OUT instruction. The following example shows how to enter a single contact and a single output coil.



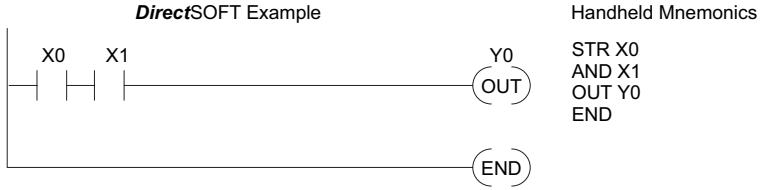
Normally Closed Contact

Normally closed contacts are also very common. This is accomplished with the Store Not, or STRN instruction. The following example shows a simple rung with a normally closed contact.



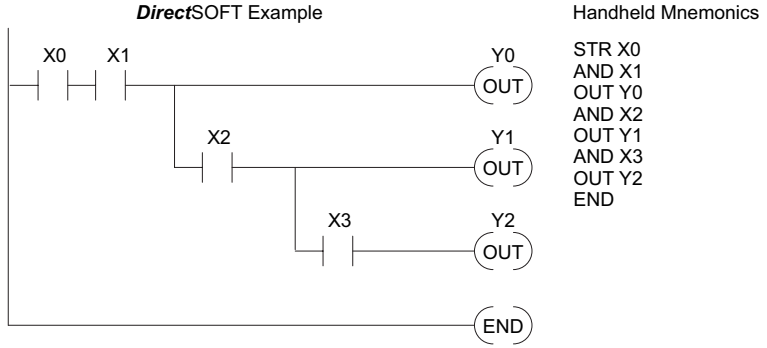
Contacts in Series

Use the AND instruction to join two or more contacts in series. The following example shows two contacts in series and a single output coil. The instructions used would be STR X0, AND X1, followed by OUT Y0.



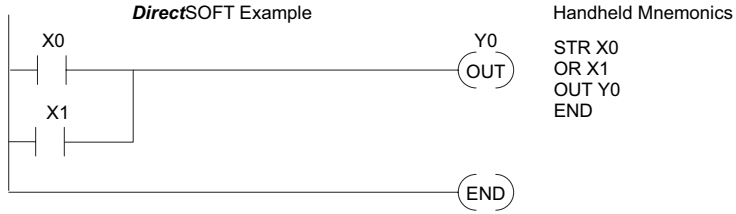
Midline Outputs

Sometimes it is necessary to use midline outputs to get additional outputs that are conditional on other contacts. The following example shows how you can use the AND instruction to continue a rung with more conditional outputs.



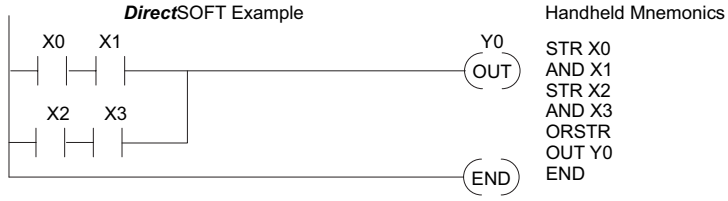
Parallel Elements

You may also have to join contacts in parallel. The OR instruction allows you to do this. The following example shows two contacts in parallel and a single output coil. The instructions would be STR X0, OR X1, followed by OUT Y0.



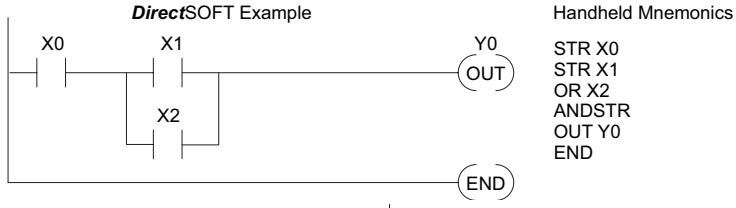
Joining Series Branches in Parallel

Quite often it is necessary to join several groups of series elements in parallel. The Or Store (ORSTR) instruction allows this operation. The following example shows a simple network consisting of series elements joined in parallel.



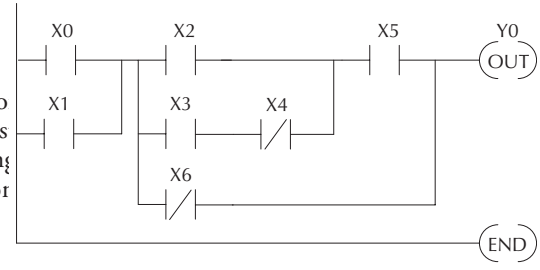
Joining Parallel Branches in Series

You can also join one or more parallel branches in series. The And Store (ANDSTR) instruction allows this operation. The following example shows a simple network with contact branches in series with parallel contacts.



Combination Networks

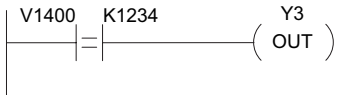
You can combine the various types o series and parallel branches to solve mos any application problem. The following example shows a simple combinatior network.



Comparative Boolean

The DL205 Micro PLCs provide Comparative Boolean instructions that allow you to quickly and easily compare two numbers. The Comparative Boolean provides evaluation of two 4-digit values using boolean contacts. The valid evaluations are: equal to, not equal to, equal to or greater than, and less than.

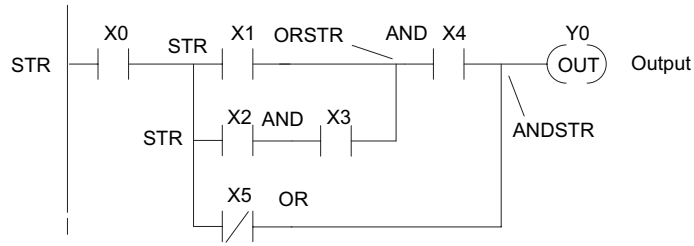
In the example ,when the BCD value in V-memory location V1400 is equal to the constant value 1234, Y3 will energize.



Boolean Stack

There are limits to how many elements you can include in a rung. This is because the DL205 CPUs use an 8-level boolean stack to evaluate the various logic elements. The boolean stack is a temporary storage area that solves the logic for the rung. Each time you enter a STR instruction, the instruction is placed on the top of the boolean stack. Any other STR instructions on the boolean stack are pushed down a level. The ANDSTR, and ORSTR instructions combine levels of the boolean stack when they are encountered. Since the boolean stack is only eight levels, an error will occur if the CPU encounters a rung that uses more than the eight levels of the boolean stack.

The following example shows how the boolean stack is used to solve boolean logic.



STR X0

1	STR X0
2	
3	
4	

STR X1

1	STR X1
2	STR X0
3	
4	

STR X2

1	STR X2
2	STR X1
3	STR X0
4	

AND X3

1	STR X2
2	STR X1
3	STR X0
4	

ORSTR

1	X1 or (X2 AND X3)
2	STR X0
3	

AND X4

1	X4 AND {X1 or (X2 AND X3)}
2	STR X0
3	

ORNOT X5

1	NOT X5 OR X4 AND {X1 OR (X2 AND X3)}
2	STR X0
3	

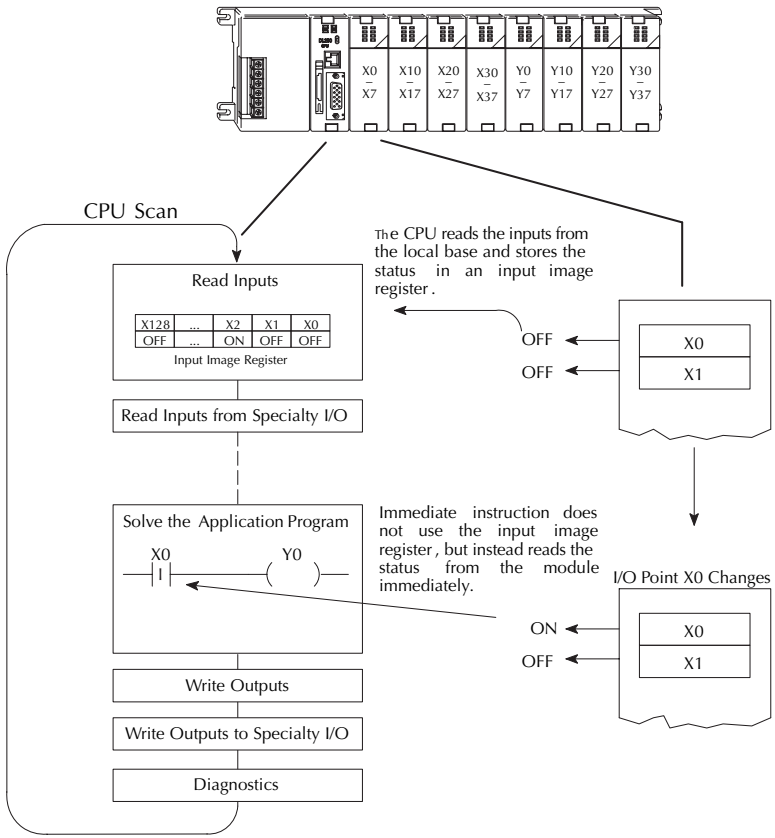
ANDSTR

1	XO AND (NOT X5 or X4) AND {X1 or (X2 AND X3)}
2	
3	

Immediate Boolean

The DL205 Micro PLCs can usually complete an operation cycle in a matter of milliseconds. However, in some applications you may not be able to wait a few milliseconds until the next I/O update occurs. The DL205 PLCs offer immediate input and outputs which are special boolean instructions that allow reading directly from inputs and writing directly to outputs during the program execution portion of the CPU cycle. You may recall that this is normally done during the input or output update portion of the CPU cycle. The immediate instructions take longer to execute because the program execution is interrupted while the CPU reads or writes the I/O point. This function is not normally done until the read inputs or the write outputs portion of the CPU cycle.

NOTE: *Even though the immediate input instruction reads the most current status from the input point, it only uses the results to solve that one instruction. It does not use the new status to update the image register. Therefore, any regular instructions that follow will still use the image register values. Any immediate instructions that follow will access the I/O again to update the status. The immediate output instruction will write the status to the I/O and update the image register.*

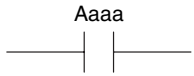


Boolean Instructions

Store (STR)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Store instruction begins a new rung or an additional branch in a rung with a normally open contact. Status of the contact will be the same state as the associated image register point or memory location.



Store Not (STRN)

The Store Not instruction begins a new rung or an additional branch in a rung with a normally closed contact. Status of the contact will be opposite the state of the associated image register point or memory location.

Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
Inputs	X	0 – 177	0 – 477	0 – 777	0 – 1777
Outputs	Y	0 – 177	0 – 477	0 – 777	0 – 1777
Control Relays	C	0 – 377	0 – 377	0 – 1777	0 – 3777
Stage	S	0 – 377	0 – 777	0 – 1777	0 – 1777
Timer	T	0 – 77	0 – 177	0 – 377	0 – 377
Counter	CT	0 – 77	0 – 177	0 – 177	0 – 377
Special Relay	SP	0 – 117, 540 – 577	0 – 137 540 – 617	0 – 777	0 – 777
Global	GX	–	–	–	0 –3777
Global	GY	–	–	–	0 – 3777

In the following Store example, when input X1 is on output Y2 will energize.

DS	Used
HPP	Used

*Direct*SOFT

Handheld Programmer Keystrokes

$ STR	→	B 1	ENT
GX OUT	→	C 2	ENT

In the following Store Not example, when input X1 is off output Y2 will energize.

*Direct*SOFT

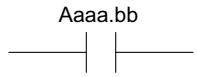
Handheld Programmer Keystrokes

SP STRN	→	B 1	ENT
GX OUT	→	C 2	ENT

Store Bit-of-Word (STRB)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

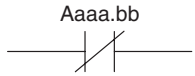
The Store Bit-of-Word instruction begins a new rung or an additional branch in a rung with a normally open contact. Status of the contact will be the same state as the bit referenced in the associated memory location.



Store Not Bit-of-Word (STRNB)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The Store Not instruction begins a new rung or an additional branch in a rung with a normally closed contact. Status of the contact will be opposite the state of the bit referenced in the associated memory location.

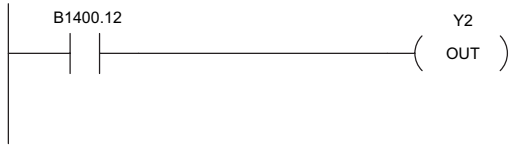


Operand Data Type		D2-250-1 Range		D2-260/D2-262 Range	
	A	aaa	bb	aaa	bb
V-memory	B	See memory map page 3-56	BCD, 0 to 15	See memory map page 3-57	BCD, 0 to 15
Pointer	PB	See memory map page 3-56	BCD, 0 to 15	See memory map page 3-57	BCD, 0 to 15

In the following Store Bit-of-Word example, when bit 12 of V-memory location V1400 is on, output Y2 will energize.

DS	Used
HPP	Used

*Direct*SOFT



Handheld Programmer Keystrokes

STR	SHFT	B	→	V	1	4	0	0
→	K	1	2	ENT				
OUT	→	2	ENT					

*Direct*SOFT

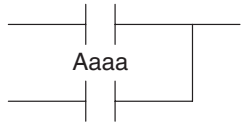
Handheld Programmer Keystrokes

STRN	SHFT	B	→	V	1	4	0	0
→	K	1	2	ENT				
OUT	→	2	ENT					

Or (OR)

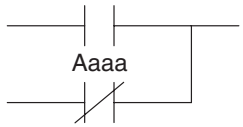
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Or instruction logically ors a normally open contact in parallel with another contact in a rung. The status of the contact will be the same state as the associated image register point or memory location.



Or Not (ORN)

The Or Not instruction logically ors a normally closed contact in parallel with another contact in a rung. The status of the contact will be opposite the state of the associated image register point or memory location.



Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
Inputs	X	0-177	0-477	0-777	0-1777
Outputs	Y	0-177	0-477	0-777	0-1777
Control Relays	C	0–377	0–377	0–1777	0–3777
Stage	S	0–377	0–777	0–1777	0–1777
Timer	T	0–77	0–177	0–377	0–377
Counter	CT	0–77	0–177	0–177	0–377
Special Relay	SP	0-117, 540-577	0-137, 540-617	0-137, 540-717	0-137, 540-717
Global	GX	-	-	-	0-3777
Global	GY	-	-	-	0-3777

In the following Or example, when input X1 or X2 is on, output Y5 will energize.

DS	Implied
HPP	Used

DirectSOFT

Handheld Programmer Keystrokes

$ STR	→	B 1	ENT
Q OR	→	C 2	ENT
GX OUT	→	F 5	ENT

In the following Or Not example, when input X1 is on or X2 is off, output Y5 will energize.

DirectSOFT

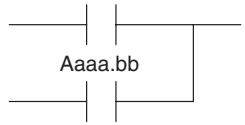
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT
R ORN	→	C 2	ENT
GX OUT	→	F 5	ENT

Or Bit-of-Word (ORB)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

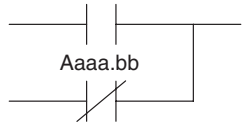
The Or Bit-of-Word instruction logically ors a normally open Bit-of-Word contact in parallel with another contact in a rung. Status of the contact will be the same state as the bit referenced in the associated memory location.



Or Not Bit-of-Word (ORNB)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

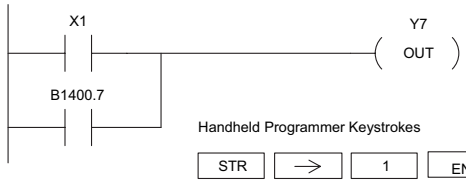
The Or Not Bit-of-Word instruction logically ors a normally closed Bit-of-Word contact in parallel with another contact in a rung. Status of the contact will be opposite the state of the bit referenced in the associated memory location.



Operand Data Type		D2-250-1 Range		D2-260/D2-262 Range	
A		aaa	bb	aaa	bb
V-memory	B	See memory map page 3-56	BCD, 0 to 15	See memory map page 3-57	BCD, 0 to 15
Pointer	PB	See memory map page 3-56	BCD, 0 to 15	See memory map page 3-57	BCD, 0 to 15

In the following Or Bit-of-Word example, when input X1 or bit 7 of V1400 is on, output Y7 will energize.

DS	Implied
HPP	Used

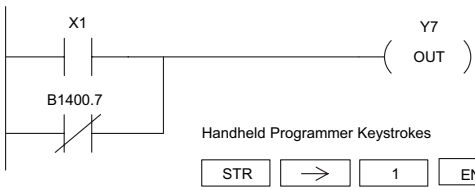


Handheld Programmer Keystrokes

STR	→	1	ENT					
OR	SHFT	B	→	V	1	4	0	0
→	K	7	ENT					
OUT	→	7	ENT					

In the following Or Not Bit-of-Word example, when input X1 is on or bit 7 of V1400 is off, output Y7 will energize.

*Direct*SOFT



Handheld Programmer Keystrokes

STR	→	1	ENT					
ORN	SHFT	B	→	V	1	4	0	0
→	K	7	ENT					
OUT	→	7	ENT					

And (AND)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The AND instruction logically ands a normally open contact in series with another contact in a rung. The status of the contact will be the same state as the associated image register point or memory location.

And Not (ANDN)

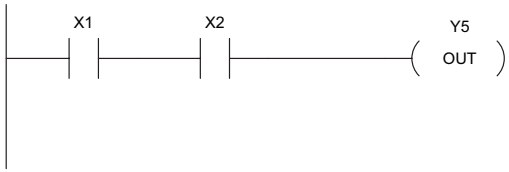
The And Not instruction logically "ANDs" a normally closed contact in series with another contact in a rung. The status of the contact will be opposite the state of the associated image register point or memory location.

Operand Data Type		D2-230	D2-240	D2-250-1	D2-260/D2-262
		Range			
	A	aaa	aaa	aaa	aaa
Inputs	X	0–177	0–477	0–777	0–1777
Outputs	Y	0–177	0–477	0–777	0–1777
Control Relays	C	0–377	0–377	0–1777	0–3777
Stage	S	0–377	0–777	0–1777	0–1777
Timer	T	0–77	0–177	0–377	0–377
Counter	CT	0–77	0–177	0–177	0–377
Special Relay	SP	0-117, 540-577	0-137, 540-617	0-137, 540-717	0-137, 540-717
Global	GX	-	-	-	0-3777
Global	GY	-	-	-	0-3777

In the following And example, when input X1 and X2 are on output Y5 will energize.

DS	Implied
HPP	Used

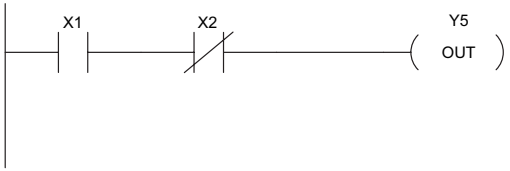
DirectSOFT



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT
V AND	→	C 2	ENT
GX OUT	→	F 5	ENT

DirectSOFT



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT
W ANDN	→	C 2	ENT
GX OUT	→	F 5	ENT

AND Bit-of-Word (ANDB)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The And Bit-of-Word instruction logically ands a normally open contact in series with another contact in a rung. The status of the contact will be the same state as the bit referenced in the associated memory location.

And Not Bit-of-Word (ANDNB)

The And Not Bit-of-Word instruction logically ands a normally closed contact in series with another contact in a rung. The status of the contact will be opposite the state of the bit referenced in the associated memory location.

Operand Data Type		D2-250-1 Range		D2-260/D2-262 Range	
	A	aaa	bb	aaa	bb
V-memory	B	See memory map page 3-56	BCD, 0 to 15	See memory map page 3-57	BCD, 0 to 15
Pointer	PB	See memory map page 3-56	BCD	See memory map page 3-57	BCD

In the following And Bit-of-Word example, when input X1 and bit 4 of V1400 is on output Y5 will energize.

DS	Implied
HPP	Used

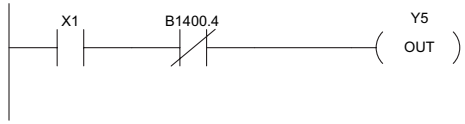
*Direct*SOFT

Handheld Programmer Keystrokes

STR	→	1	ENT					
AND	SHFT	B	→	V	1	4	0	0
→	K	4	ENT					
OUT	→	5	ENT					

In the following And Not Bit-of-Word example, when input X1 is on and bit 4 of V1400 is off, output Y5 will energize.

*Direct*SOFT



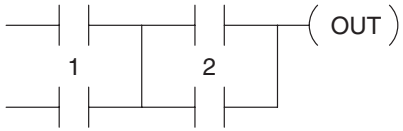
Handheld Programmer Keystrokes

STR	→	1	ENT					
ANDN	SHFT	B	→	V	1	4	0	0
→	K	4	ENT					
OUT	→	5	ENT					

And Store (ANDSTR)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

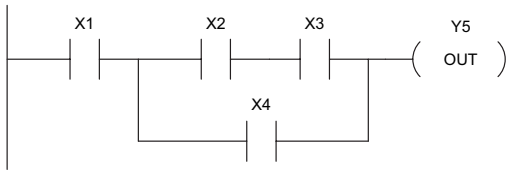
The And Store instruction logically ands two branches of a rung in series. Both branches must begin with the Store instruction.



In the following And Store example, the branch consisting of contacts X2, X3, and X4 have been anded with the branch consisting of contact X1.

DS	Implied
HPP	Used

*Direct*SOFT



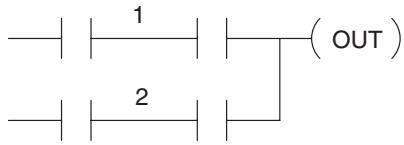
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT
$ STR	→	C 2	ENT
V AND	→	D 3	ENT
Q OR	→	E 4	ENT
L ANDST	ENT		
GX OUT	→	F 5	ENT

Or Store (ORSTR)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

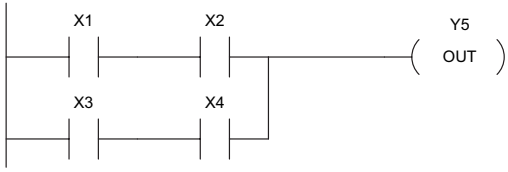
The Or Store instruction logically ors two branches of a rung in parallel. Both branches must begin with the Store instruction.



In the following Or Store example, the branch consisting of X1 and X2 have been OR'd with the branch consisting of X3 and X4.

DS	Implied
HPP	Used

*Direct*SOFT



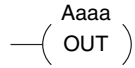
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT
V AND	→	C 2	ENT
$ STR	→	D 3	ENT
V AND	→	E 4	ENT
M ORST	ENT		
GX OUT	→	F 5	ENT

Out (OUT)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Out instruction reflects the status of the rung (on/off) and outputs the discrete (on/off) state to the specified image register point or memory location. Multiple Out instructions referencing the same discrete location should not be used since only the last Out instruction in the program will control the physical output point. Instead, use the next instruction, the Or Out.



Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
Inputs	X	0–177	0–477	0–777	0–1777
Outputs	Y	0–177	0–477	0–777	0–1777
Control Relays	C	0–377	0–377	0–1777	0–3777
Global	GX	-	-	-	0–3777
Global	GY	-	-	-	0–3777

In the following Out example, when input X1 is on, output Y2 and Y5 will energize.

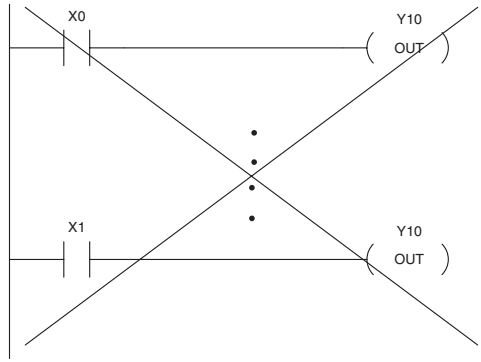
DS	Used
HPP	Used

*Direct*SOFT

Handheld Programmer Keystrokes

$ STR	→	B 1	ENT
GX OUT	→	C 2	ENT
GX OUT	→	F 5	ENT

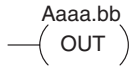
In the following Out example, the program contains two Out instructions using the same location (Y10). The physical output of Y10 is ultimately controlled by the last rung of logic referencing Y10. X1 will override the Y10 output being controlled by X0. To avoid this situation, multiple outputs using the same location should not be used in programming. If you need to have an output controlled by multiple inputs, see the OROUT instruction on page 5–19.



Out Bit-of-Word (OUTB)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

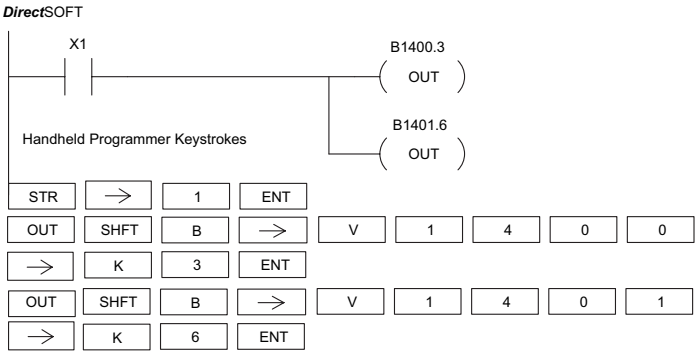
The Out Bit-of-Word instruction reflects the status of the rung (on/off) and outputs the discrete (on/off) state to the specified bit in the referenced memory location. Multiple Out Bit-of-Word instructions referencing the same bit of the same word generally should not be used since only the last Out instruction in the program will control the status of the bit.



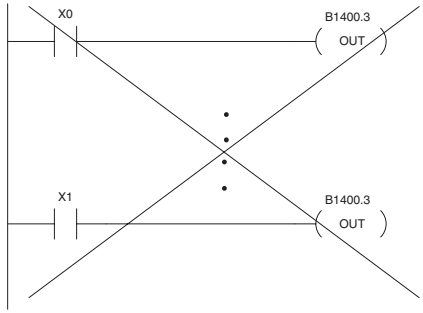
Operand Data Type		D2-250-1 Range		D2-260/D2-262 Range	
	A	aaa	bb	aaa	bb
V-memory	B	See memory map page 3-56	BCD, 0 to 15	See memory map page 3-57	BCD, 0 to 15
Pointer	PB	See memory map page 3-56	BCD	See memory map page 3-57	BCD

In the following Out Bit-of-Word example, when input X1 is on, bit 3 of V1400 and bit 6 of V1401 will turn on.

DS	Used
HPP	Used



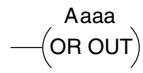
The following Out Bit-of-Word example contains two Out Bit-of-Word instructions using the same bit in the same memory word. The final state bit 3 of V1400 is ultimately controlled by the last rung of logic referencing it. X1 will override the logic state controlled by X0. To avoid this situation, multiple outputs using the same location must not be used in programming.



Or Out (OROUT)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Or Out instruction allows more than one rung of discrete logic to control a single output. Multiple Or Out instructions referencing the same output coil may be used, since *all* contacts controlling the output are logically OR'd together. If the status of *any* rung is on, the output will also be on.

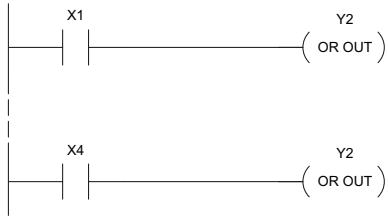


Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
Inputs	X	0–177	0–477	0–777	0–1777
Outputs	Y	0-177	0-477	0-777	0-1777
Control Relays	C	0–377	0–377	0–1777	0–3777
Global	GX	-	-	-	0–3777
Global	GY	-	-	-	0–3777

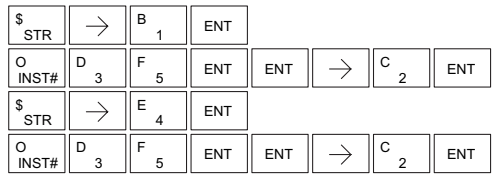
In the following example, when X1 or X4 is on, Y2 will energize.

DS	Used
HPP	Used

*Direct*SOFT



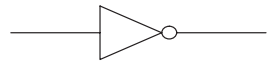
Handheld Programmer Keystrokes



Not (NOT)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

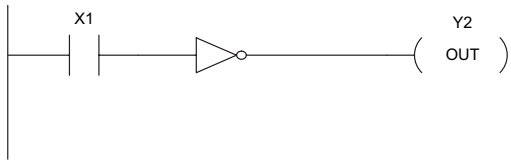
The Not instruction inverts the status of the rung at the point of the instruction.



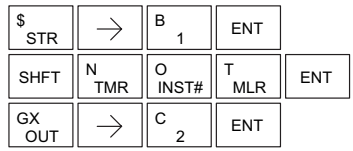
In the following example, when X1 is off, Y2 will energize. This is because the Not instruction inverts the status of the rung at the Not instruction.

DS	Used
HPP	Used

*Direct*SOFT



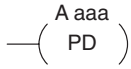
Handheld Programmer Keystrokes



☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

Positive Differential (PD)

The Positive Differential instruction is typically known as a one shot. When the input logic produces an off-to-on transition, the output will energize for one CPU scan.



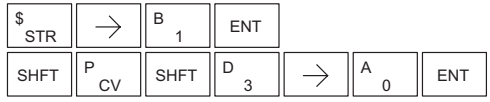
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
Inputs	X	0–177	0–477	0–777	0–1777
Outputs	Y	0–177	0–477	0–777	0–1777
Control Relays	C	0–377	0–377	0–1777	0–3777

In the following example, every time X1 makes an off to on transition, C0 will energize for one scan.

DS	Used
HPP	Used

*Direct*SOFT

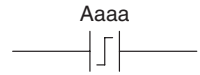
Handheld Programmer Keystrokes



NOTE: *To generate a "one–shot" pulse on an on–to–off transition, place a NOT instruction immediately before the PD instruction. The D2-250–1, D2-260 and D2-262 CPUs support the STRND instruction.*

Store Positive Differential (STRPD)

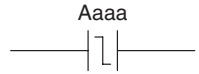
☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262



The Store Positive Differential instruction begins a new rung or an additional branch in a rung with a contact. The contact closes for one CPU scan when the state of the associated image register point makes an off-to-on transition. Thereafter, the contact remains open until the next off-to-on transition (the symbol inside the contact represents the transition). This function is sometimes called a "one-shot." 'This contact will also close on a program-to-run transition if it is within a retentive range and on before the PLC mode transition.

Store Negative Differential (STRND)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262



The Store Negative Differential instruction begins a new rung or an additional branch in a rung with a contact. The contact closes for one CPU scan when the state of the associated image register point makes an on-to-off transition. Thereafter, the contact remains open until the next on-to-off transition (the symbol inside the contact represents the transition).

Operand Data Type		D2-250-1 Range	D2-260/D2-262 Range
	A	**aaa**	**aaa**
Inputs	X	0–777	0–1777
Outputs	Y	0–777	0–1777
Control Relays	C	0–1777	0–3777
Stage	S	0–1777	0–1777
Timer	T	0–377	0–377
Counter	CT	0–177	0–377
Global	GX	–	0–3777
Global	GY	–	0–3777

In the following example, each time X1 is makes an off-to-on transition, Y4 will energize for one scan.

DS	Used
HPP	Used

*Direct*SOFT

Handheld Programmer Keystrokes

$ STR	SHFT	P CV	D 3	→	B 1	ENT
GX OUT	→	E 4	ENT			

In the following example, each time X1 makes an on-to-off transition, Y4 will energize for one scan.

*Direct*SOFT

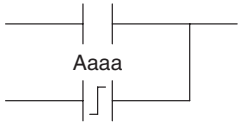
Handheld Programmer Keystrokes

$ STR	SHFT	N TMR	D 3	→	B 1	ENT
GX OUT	→	E 4	ENT			

Or Positive Differential (ORPD)

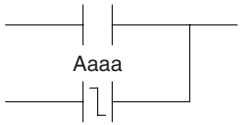
☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The Or Positive Differential instruction logically ORs a contact in parallel with another contact in a rung. The status of the contact will be open until the associated image register point makes an off-to-on transition, closing it for one CPU scan. Thereafter, it remains open until another off-to-on transition.



Or Negative Differential (ORND)

The Or Negative Differential instruction logically ORs a contact in parallel with another contact in a rung. The status of the contact will be open until the associated image register point makes an on-to-off transition, closing it for one CPU scan. Thereafter, it remains open until another on-to-off transition.



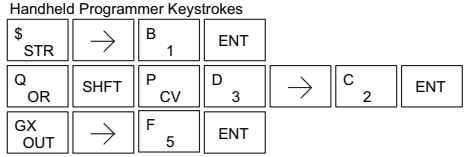
Operand Data Type		D2-250-1 Range	D2-260/D2-262 Range
	A	aaa	aaa
Inputs	X	0–777	0–1777
Outputs	Y	0–777	0–1777
Control Relays	C	0–1777	0–3777
Stage	S	0–1777	0–1777
Timer	T	0–377	0–377
Counter	CT	0–177	0–377
Global	GX	–	0–3777
Global	GY	–	0–3777

In the following example, Y5 will energize whenever X1 is on, or for one CPU scan when X2 transitions from off to on.

DS	Implied
HPP	Used

*Direct*SOFT

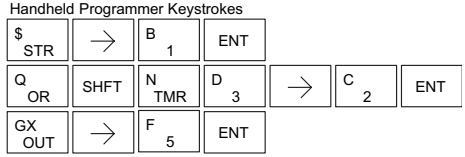
Handheld Programmer Keystrokes



In the following example, Y5 will energize whenever X1 is on, or for one CPU scan when X2 transitions from on to off.

*Direct*SOFT

Handheld Programmer Keystrokes



And Positive Differential (ANDPD)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The And Positive Differential instruction logically ANDs a normally open Positive Differential contact in series with another contact in a rung. The status of the contact will be open until the associated image register point makes an off-to-on transition, closing it for one CPU scan. Thereafter, it remains open until another off-to-on transition.

And Negative Differential (ANDND)

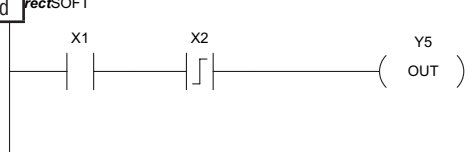
The And Negative Differential instruction logically ANDs a normally open Negative Differential contact in series with another contact in a rung. The status of the contact will be open until the associated image register point makes an on-to-off transition, closing it for one CPU scan. Thereafter, it remains open until another on-to-off transition.

Operand Data Type	D2-250-1 Range	D2-260/D2-262 Range
	A	**aaa**	**aaa**
Inputs	X	0–777	0–1777
Outputs	Y	0–777	0–1777
Control Relays	C	0–1777	0–3777
Stage	S	0–1777	0–1777
Timer	T	0–377	0–377
Counter	CT	0–177	0–377
Global	GX	–	0–3777
Global	GY	–	0–3777

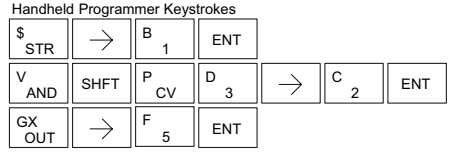
In the following example, Y5 will energize for one CPU scan whenever X1 is on and X2 transitions from off to on.

DS	Implied
HPP	Used

*Direct*SOFT

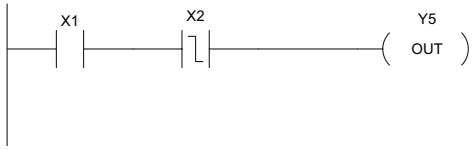


Handheld Programmer Keystrokes

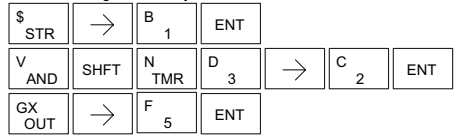


In the following example, Y5 will energize for one CPU scan whenever X1 is on and X2 transitions from on to off.

*Direct*SOFT



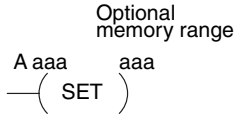
Handheld Programmer Keystrokes



Set (SET)

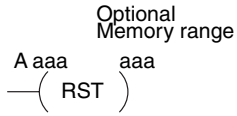
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Set instruction sets or turns on an image register point/memory location or a consecutive range of image register points/memory locations. Once the point/location is set, it will remain on until it is reset using the Reset instruction. It is not necessary for the input controlling the Set instruction to remain on.



Reset (RST)

The Reset instruction resets or turns off an image register point/memory location or a range of image registers points/memory locations. Once the point/location is reset, it is not necessary for the input to remain on.



Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
Inputs	X	0–177	0–477	0–777	0–1777
Outputs	Y	0-177	0-477	0-777	0-1777
Control Relays	C	0–377	0–377	0–1777	0–3777
Stage	S	0-377	0-777	0-1777	0-1777
Timer*	T	0-77	0-177	0-377	0-377
Counter *	CT	0-77	0-177	0-177	0-377
Global	GX	-	-	-	0–3777
Global	GY	-	-	-	0–3777
* Timer and counter operand data types are not valid using the Set instruction					

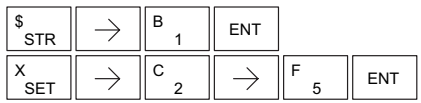
NOTE: *You cannot set inputs (Xs) that are assigned to input modules*

In the following example, when X1 is on, Y2 through Y5 will energize.

DS	Used
HPP	Used

DirectSOFT

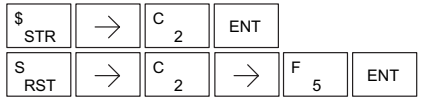
Handheld Programmer Keystrokes



In the following example, when X2 is on, Y2 through Y5 will be reset or de–energized.

DirectSOFT

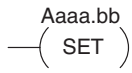
Handheld Programmer Keystrokes



Set Bit-of-Word (SETB)

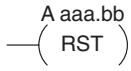
☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The Set Bit-of-Word instruction sets or turns on a bit in a V-memory location. Once the bit is set, it will remain on until it is reset using the Reset Bit-of-Word instruction. It is not necessary for the input controlling the Set Bit-of-Word instruction to remain on.



Reset Bit-of-Word (RSTB)

The Reset Bit-of-Word instruction resets or turns off a bit in a V-memory location. Once the bit is reset, it is not necessary for the input to remain on.



Operand Data Type		D2-250-1 Range		D2-260/D2-262 Range	
	A	aaa	bb	aaa	bb
V-memory	B	See memory map	BCD, 0 to 15	See memory map	BCD, 0 to 15
Pointer	PB	See memory map	BCD	See memory map	BCD

In the following example, when X1 turns on, bit 1 in V1400 is set to the on state.

DS	Used
HPP	Used

*Direct*SOFT

Handheld Programmer Keystrokes

STR	→	1	ENT					
SET	SHFT	B	→	V	1	4	0	0
→	K	1	ENT					

In the following example, when X2 turns on, bit 1 in V1400 is reset to the off state.

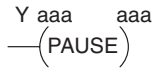
*Direct*SOFT

Handheld Programmer Keystrokes

STR	→	2	ENT					
RST	SHFT	B	→	V	1	4	0	0
→	K	1	ENT					

☑ 230 ☑ 240 ☑ 250-1 ☑ 260 ☑ 262 Pause (PAUSE)

The Pause instruction disables the output update on a range of outputs. The ladder program will continue to run and update the image register; however, the outputs in the range specified in the Pause instruction will be turned off at the output points.

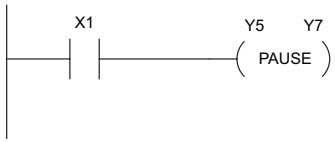


Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
Outputs	Y	0-177	0-477	0-777	0-1777

In the following example, when X1 is ON, Y5–Y7 will be turned OFF. The execution of the ladder program will not be affected.

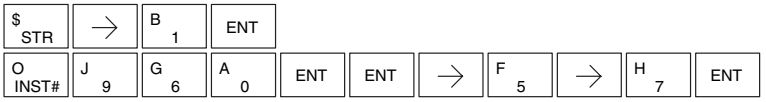
DS	Used
HPP	Used

DirectSOFT



Since the D2–HPP Handheld Programmer does not have a specific Pause key, you can use the corresponding instruction number for entry (#960) or type each letter of the command.

Handheld Programmer Keystrokes



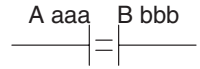
In some cases, you may want certain output points in the specified pause range to operate normally. In that case, use Aux 58 to override the Pause instruction.

Comparative Boolean

Store If Equal (STRE)

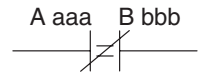
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Store If Equal instruction begins a new rung or additional branch in a rung with a normally open comparative contact. The contact will be ON when Aaaa equals Bbbb .



Store If Not Equal (STRNE)

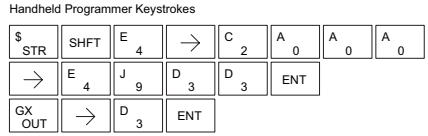
The Store If Not Equal instruction begins a new rung or additional branch in a rung with a normally closed comparative contact. The contact will be ON when Aaaa does not equal Bbbb.



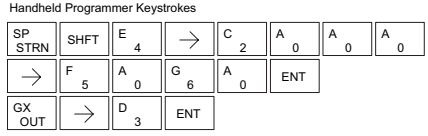
Operand Data Type		Range							
		D2-230		D2-240		D2-250-1		D2-260/D2-262	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All. (See memory map page 3-54)	All. (See memory map page 3-54)	All. (See memory map page 3-55)	All. (See memory map page 3-55)	All. (See memory map page 3-55)	All. (See memory map page 3-56)	All. (See memory map page 3-57)	All. (See memory map page 3-57)
Pointer	P	–	–	–	All. (See memory map page 3-55)	–	All. (See memory map page 3-56)	–	All. (See memory map page 3-57)
Constant	K	–	0-FFFF	–	0-FFFF	–	0-FFFF	–	0-FFFF

In the following example, when the value in V-memory location V2000 = 4933 , Y3 will energize.

DS	Implied
HPP	Used



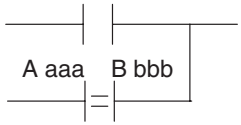
In the following example, when the value in V-memory location V2000 ≠ 5060, Y3 will energize.



Or If Equal (ORE)

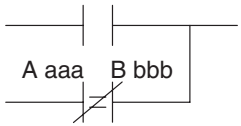
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Or If Equal instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa equals Bbbb.



Or If Not Equal (ORNE)

The Or If Not Equal instruction connects a normally closed comparative contact in parallel with another contact. The contact will be on when Aaaa does not equal Bbbb.

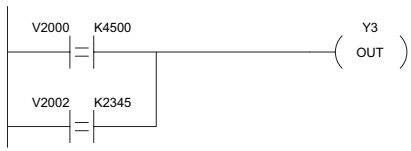


Operand Data Type		Range							
		D2-230		D2-240		D2-250-1		D2-260/D2-262	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All. (See memory map page 3-54)	All. (See memory map page 3-54)	All. (See memory map page 3-55)	All. (See memory map page 3-55)	All. (See memory map page 3-56)	All. (See memory map page 3-56)	All. (See memory map page 3-57)	All. (See memory map page 3-57)
Pointer	P	–	–	–	All. (See memory map page 3-55)	–	All. (See memory map page 3-56)	–	All. (See memory map page 3-57)
Constant	K	–	0-FFFF	–	0-FFFF	–	0-FFFF	–	0-FFFF

In the following example, when the value in V-memory location V2000 = 4500 or V2202 = 2345, Y3 will energize.

DS	Implied
HPP	Used

*Direct*SOFT

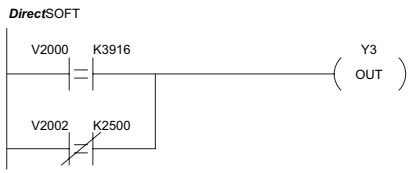


Handheld Programmer Keystrokes

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
E 4	F 5	A 0	A 0	ENT				
Q OR	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

In the following example, when the value in V-memory location V2000 = 3916 or V2002 ≠ 2500, Y3 will energize.

*Direct*SOFT



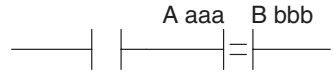
Handheld Programmer Keystrokes

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
D 3	J 9	B 1	G 6	ENT				
R ORN	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	D 3	ENT					

And If Equal (ANDE)

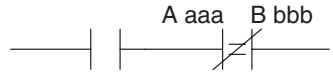
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The And If Equal instruction connects a normally open comparative contact in series with another contact. The contact will be on when Aaaa equals Bbbb.



And If Not Equal (ANDNE)

The And If Not Equal instruction connects a normally closed comparative contact in series with another contact. The contact will be on when Aaaa does not equal Bbbb



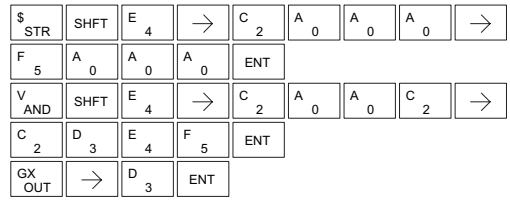
Operand Data Type		Range							
		D2-230		D2-240		D2-250-1		D2-260/D2-262	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All. (See memory map page 3-54)	All. (See memory map page 3-54)	All. (See memory map page 3-55)	All. (See memory map page 3-55)	All. (See memory map page 3-56)	All. (See memory map page 3-56)	All. (See memory map page 3-57)	All. (See memory map page 3-57)
Pointer	P	–	–	–	All. (See memory map page 3-55)	–	All. (See memory map page 3-56)	–	All. (See memory map page 3-57)
Constant	K	–	0-FFFF	–	0-FFFF	–	0-FFFF	–	0-FFFF

In the following example, when the value in V-memory location V2000 = 5000 and V2002 = 2345, Y3 will energize.

DS	Implied
HPP	Used

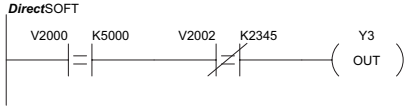
*Direct*SOFT

Handheld Programmer Keystrokes

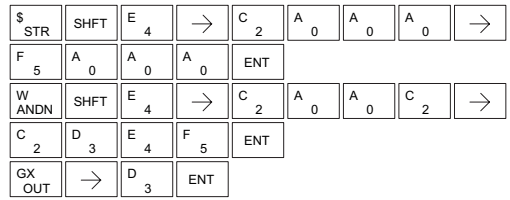


In the following example, when the value in V-memory location V2000 = 5000 and V2002 ≠ 2345, Y3 will energize.

*Direct*SOFT



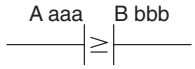
Handheld Programmer Keystrokes



Store (STR)

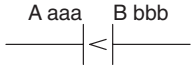
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Comparative Store instruction begins a new rung or additional branch in a rung with a normally open comparative contact. The contact will be on when Aaaa is equal to or greater than Bbbb.



Store Not (STRN)

The Comparative Store Not instruction begins a new rung or additional branch in a rung with a normally open comparative contact. The contact will be on when Aaaa is less than Bbbb.



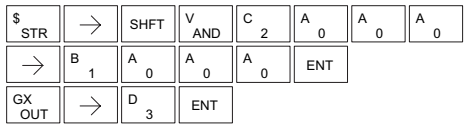
Operand Data Type		Range							
		D2-230		D2-240		D2-250-1		D2-260/D2-262	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All. (See memory map page 3-54)	All. (See memory map page 3-54)	All. (See memory map page 3-55)	All. (See memory map page 3-55)	All. (See memory map page 3-56)	All. (See memory map page 3-56)	All. (See memory map page 3-57)	All. (See memory map page 3-57)
Pointer	P	–	–	–	All. (See memory map page 3-55)	–	All. (See memory map page 3-56)	–	All. (See memory map page 3-57)
Constant	K	–	0-FFFF	–	0-FFFF	–	0-FFFF	–	0-FFFF

In the following example, when the value in V-memory location V2000 ≥ 1000, Y3 will energize.

DS	Implied
HPP	Used

*Direct*SOFT

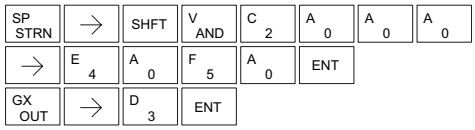
Handheld Programmer Keystrokes



In the following example, when the value in V-memory location V2000 < 4050, Y3 will energize.

*Direct*SOFT

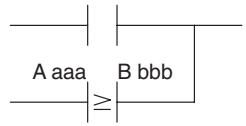
Handheld Programmer Keystrokes



Or (OR)

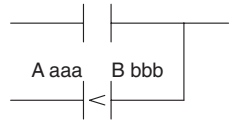
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Comparative Or instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa is equal to or greater than Bbbb.



Or Not (ORN)

The Comparative Or Not instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa is less than Bbbb.

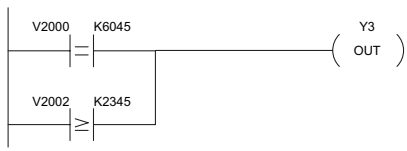


Operand Data Type		Range							
		D2-230		D2-240		D2-250-1		D2-260/D2-262	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All. (See memory map page 3-54)	All. (See memory map page 3-54)	All. (See memory map page 3-55)	All. (See memory map page 3-55)	All. (See memory map page 3-56)	All. (See memory map page 3-56)	All. (See memory map page 3-57)	All. (See memory map page 3-57)
Pointer	P	–	–	–	All. (See memory map page 3-55)	–	All. (See memory map page 3-56)	–	All. (See memory map page 3-57)
Constant	K	–	0-FFFF	–	0-FFFF	–	0-FFFF	–	0-FFFF

In the following example, when the value in V-memory location V2000 = 6045 or V2002 ≥ 2345, Y3 will energize.

DS	Implied
HPP	Used

*Direct*SOFT

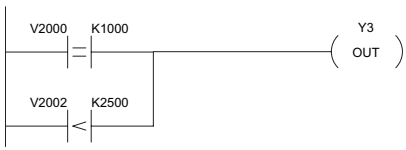


Handheld Programmer Keystrokes

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
G 6	A 0	E 4	F 5	ENT				
Q OR	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

In the following example when the value in V-memory location V2000 = 1000 or V2002 < 2500, Y3 will energize.

*Direct*SOFT



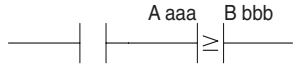
Handheld Programmer Keystrokes

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
B 1	A 0	A 0	A 0	ENT				
R ORN	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	D 3	ENT					

And (AND)

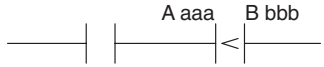
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Comparative And instruction connects a normally open comparative contact in series with another contact. The contact will be on when Aaaa is equal to or greater than Bbbb.



And Not (ANDN)

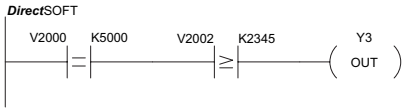
The Comparative And Not instruction connects a normally open comparative contact in series with another contact. The contact will be on when Aaaa < Bbbb.



Operand Data Type		Range							
		D2-230		D2-240		D2-250-1		D2-260/D2-262	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All. (See memory map page 3-54)	All. (See memory map page 3-54)	All. (See memory map page 3-55)	All. (See memory map page 3-55)	All. (See memory map page 3-56)	All. (See memory map page 3-56)	All. (See memory map page 3-57)	All. (See memory map page 3-57)
Pointer	P	–	–	–	All. (See memory map page 3-55)	–	All. (See memory map page 3-56)	–	All. (See memory map page 3-57)
Constant	K	–	0-FFFF	–	0-FFFF	–	0-FFFF	–	0-FFFF

In the following example, when the value in V-memory location V2000 = 5000, and V2002 ≥ 2345, Y3 will energize.

DS	Implied
HPP	Used



Handheld Programmer Keystrokes

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
F 5	A 0	A 0	A 0	ENT				
V AND	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

In the following example, when the value in V-memory location V2000 = 7000 and V2002 < 2500, Y3 will energize.

Handheld Programmer Keystrokes

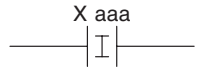
$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
H 7	A 0	A 0	A 0	ENT				
W ANDN	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	D 3	ENT					

Immediate Instructions

Store Immediate (STRI)

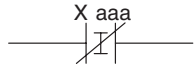
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Store Immediate instruction begins a new rung or additional branch in a rung. The status of the contact will be the same as the status of the associated input point *at the time the instruction is executed*. The image register is not updated.



Store Not Immediate (STRNI)

The Store Not Immediate instruction begins a new rung or additional branch in a rung. The status of the contact will be opposite the status of the associated input point *at the time the instruction is executed*. The image register is not updated.



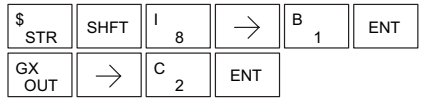
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
Inputs	X	0–177	0–477	0–777	0–1777

In the following example, when X1 is on, Y2 will energize.

DS	Implied
HPP	Used

***Direct*SOFT**

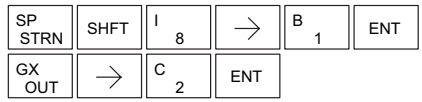
Handheld Programmer Keystrokes



In the following example, when X1 is off, Y2 will energize.

***Direct*SOFT**

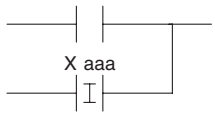
Handheld Programmer Keystrokes



Or Immediate (ORI)

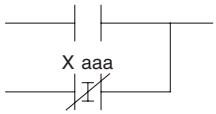
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Or Immediate connects two contacts in parallel. The status of the contact will be the same as the status of the associated input point *at the time the instruction is executed.* The image register is not updated.



Or Not Immediate (ORNI)

The Or Not Immediate connects two contacts in parallel. The status of the contact will be opposite the status of the associated input point *at the time the instruction is executed.* The image register is not updated.

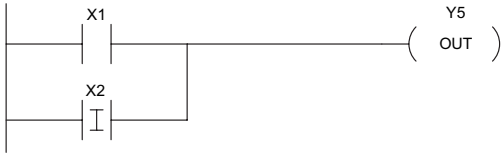


Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
Inputs	X	0–177	0–477	0–777	0–1777

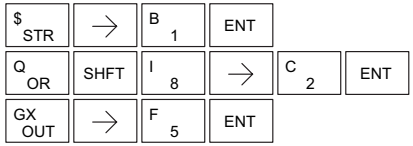
In the following example, when X1 or X2 is on, Y5 will energize.

DS	Implied
HPP	Used

***Direct*SOFT**

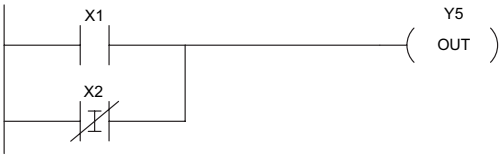


Handheld Programmer Keystrokes

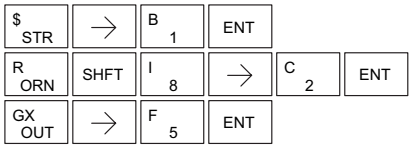


In the following example, when X1 is on or X2 is off, Y5 will energize.

***Direct*SOFT**



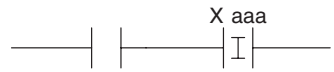
Handheld Programmer Keystrokes



And Immediate (ANDI)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The And Immediate connects two contacts in series. The status of the contact will be the same as the status of the associated input point *at the time the instruction is executed*. The image register is not updated.



And Not Immediate (ANDNI)

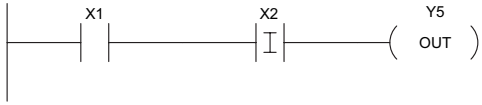
The And Not Immediate connects two contacts in series. The status of the contact will be opposite the status of the associated input point *at the time the instruction is executed*. The image register is not updated.

Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
Inputs	X	0–177	0–477	0–777	0–1777

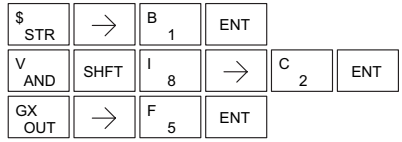
In the following example, when X1 and X2 are on, Y5 will energize.

DS	Implied
HPP	Used

***Direct*SOFT**

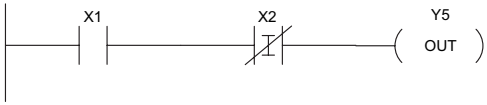


Handheld Programmer Keystrokes

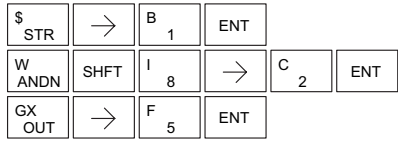


In the following example, when X1 is on and X2 is off, Y5 will energize.

***Direct*SOFT**



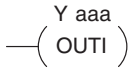
Handheld Programmer Keystrokes



Out Immediate (OUTI)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

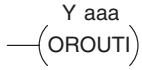
The Out Immediate instruction reflects the status of the rung (on/off) and outputs the discrete (on/off) status to the specified module output point and the image register *at the time the instruction is executed*. If multiple Out Immediate instructions referencing the same discrete point are used, it is possible for the module output status to change multiple times in a CPU scan. See Or Out Immediate.



Or Out Immediate (OROUTI)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Or Out Immediate instruction has been designed to use more than one rung of discrete logic to control a single output. Multiple Or Out Immediate instructions referencing the same output coil may be used, since all contacts controlling the output are ORed together. If the status of any rung is on *at the time the instruction is executed*, the output will also be on.



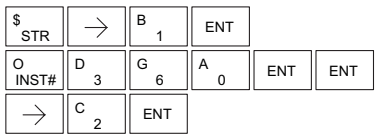
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
Outputs	Y	0–177	0–477	0–777	0–1777

In the following example, when X1 is on, output point Y2 on the output module will turn on. For instruction entry on the Handheld Programmer, you can use the instruction number (#350) as shown, or type each letter of the command.

DS	Used
HPP	Used

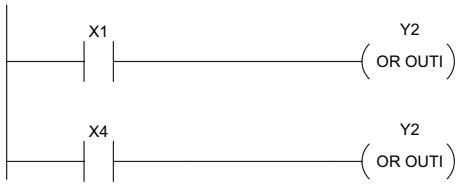
DirectSOFT

Handheld Programmer Keystrokes

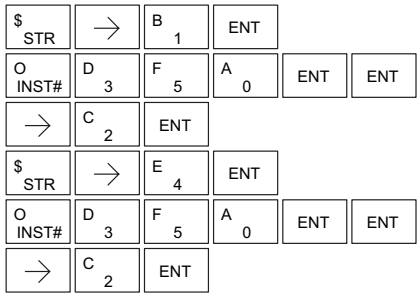


In the following example, when X1 or X4 is on, Y2 will energize.

DirectSOFT



Handheld Programmer Keystrokes



Out Immediate Formatted (OUTIF)

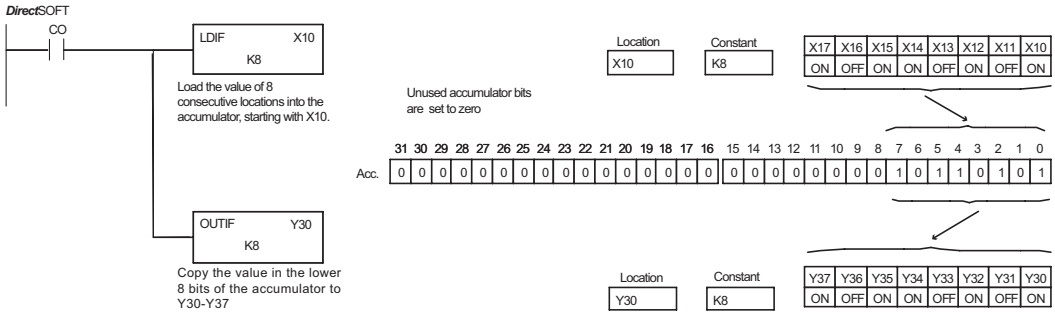
☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

The Out Immediate Formatted instruction outputs a 1 to 32 bit binary value from the accumulator to specified output points *at the time the instruction is executed*. Accumulator bits that are not used by the instruction are set to zero.

Operand Data Type		D2-260/D2-262 Range	
		aaa	bbb
Outputs	Y	0–1777	–
Constant	K	–	1–32

In the following example, when C0 is on,the binary pattern for X10 –X17 is loaded into the accumulator using the Load Immediate Formatted instruction. The binary pattern in the accumulator is written to Y30–Y37 using the Out Immediate Formatted instruction. This technique is useful to quickly copy an input pattern to outputs (without waiting for the CPU scan).

DS	Used
HPP	Used



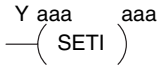
Handheld Programmer Keystrokes

$ STR	→	NEXT	NEXT	NEXT	NEXT	A 0	ENT				
SHFT	L ANDST	D 3	I 8	F 5	→	B 1	A 0	→	I 8	ENT	
GX OUT	SHFT	I 8	F 5	→	D 3	A 0	→	I 8	ENT		

Set Immediate (SETI)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

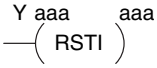
The Set Immediate instruction immediately sets or turns on an output or a range of outputs in the image register and the corresponding output point(s) *at the time the instruction is executed.* Once the outputs are set, it is not necessary for the input to remain on. The Reset Immediate instruction can be used to reset the outputs.



Reset Immediate (RSTI)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Reset Immediate instruction immediately resets or turns off an output or a range of outputs in the image register and the output point(s) *at the time the instruction is executed.* Once the outputs are reset, it is not necessary for the input to remain on.



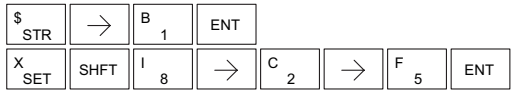
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
Outputs	Y	0–177	0–477	0–777	0–1777

In the following example, when X1 is on, Y2 through Y5 will be set on in the image register and on the corresponding output points.

DS	Used
HPP	Used

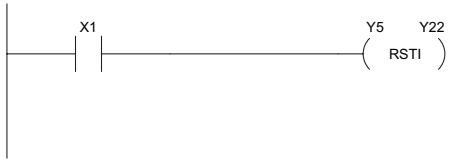
*Direct*SOFT

Handheld Programmer Keystrokes

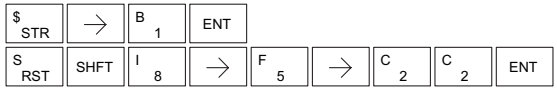


In the following example, when X1 is on, Y5 through Y22 will be reset (off) in the image register and on the corresponding output module(s).

*Direct*SOFT



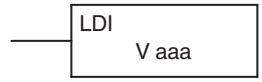
Handheld Programmer Keystrokes



Load Immediate (LDI)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

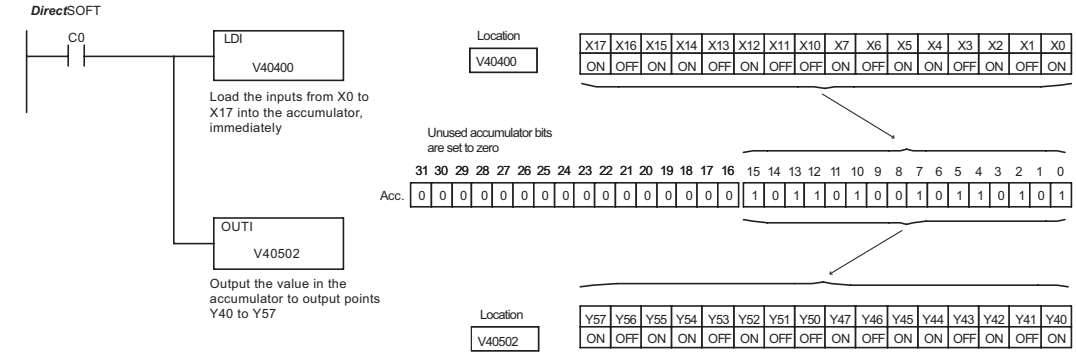
The Load Immediate instruction loads a 16-bit V-memory value into the accumulator. The valid address range includes all input point addresses on the local base. The value reflects the current status of the input points *at the time the instruction is executed*. This instruction may be used instead of the LDIF instruction, which requires you to specify the number of input points.



Operand Data Type		D2-260/D2-262 Range
		aaaaa
Inputs V-memory	V	40400–40477

In the following example, when C0 is on, the binary pattern of X0–X17 will be loaded into the accumulator using the Load Immediate instruction. The Out Immediate instruction could be used to copy the 16 bits in the accumulator to output points, such as Y40–Y57. This technique is useful to quickly copy an input pattern to output points (without waiting for a full CPU scan to occur).

DS	Used
HPP	Used



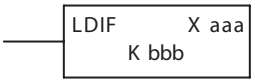
Handheld Programmer Keystrokes

$ STR	→	SHFT	C 2	A 0	ENT						
SHFT	L ANDST	D 3	I 8	→	E 4	A 0	E 4	A 0	A 0	ENT	
GX OUT	SHFT	I 8	→	NEXT	E 4	A 0	F 5	A 0	C 2	ENT	

Load Immediate Formatted (LDIF)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

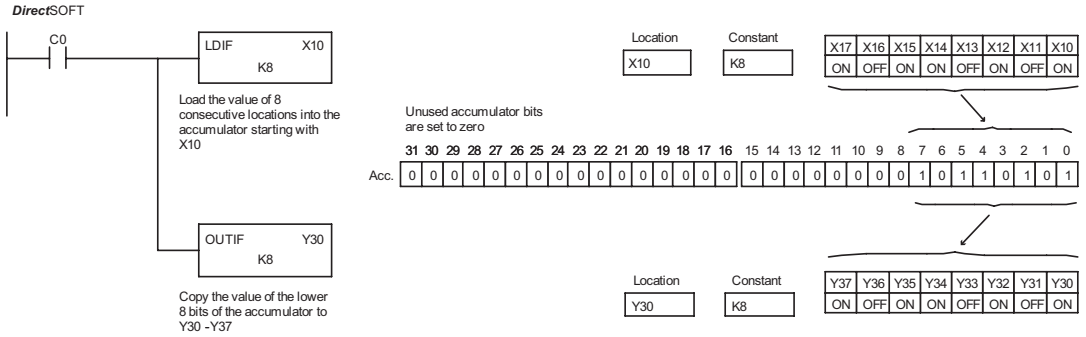
The Load Immediate Formatted instruction loads a 1–32 bit binary value into the accumulator. The value reflects the current status of the input module(s) *at the time the instruction is executed*. Accumulator bits that are not used by the instruction are set to zero.



Operand Data Type		D2-260/D2-262 Range	
		aaa	bbb
Intputs	X	0–1777	–
Constant	K	–	1–32

In the following example, when C0 is on, the binary pattern of X10–X17 will be loaded into the accumulator using the Load Immediate Formatted instruction. The Out Immediate Formatted instruction could be used to copy the specified number of bits in the accumulator to the specified outputs on the output module, such as Y30–Y37. This technique is useful to quickly copy an input pattern to outputs (without waiting for the CPU scan).

DS	Used
HPP	Used



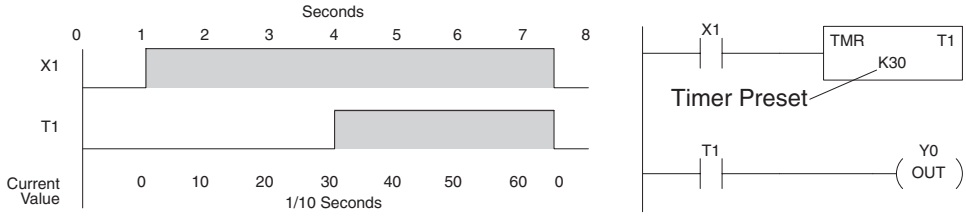
Handheld Programmer Keystrokes

$ STR	→	SHFT	C 2	A 0	ENT							
SHFT	L ANDST	D 3	I 8	F 5	→	B 1	A 0	→	I 8	ENT		
GX OUT	SHFT	I 8	F 5	→	D 3	A 0	→	I 8	ENT			

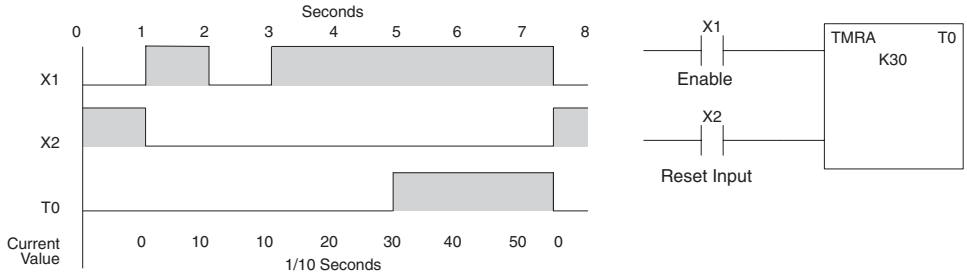
Timer, Counter and Shift Register Instructions

Using Timers

Timers are used to time an event for a desired length of time. The single input timer (TMR) will time as long as the input is on. When the input changes from on to off, the timer current value is reset to 0. There is a tenth of a second and a hundredth of a second timer available with a maximum time of 999.9 and 99.99 seconds respectively. A discrete bit is associated with each timer to indicate that the current value is equal to or greater than the preset value. The timing diagram below shows the relationship between the timer input, associated discrete bit, current value, and timer preset.



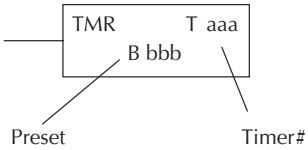
Some applications that need an accumulating timer, meaning it has the ability to time, stop, and then resume from where it previously stopped. The accumulating timer (TMRA) works similarly to the regular timer, but two inputs are required. The enable input starts and stops the timer. When the timer stops, the elapsed time is maintained. When the timer starts again, the timing continues from the elapsed time. When the reset input is turned on, the elapsed time is cleared and the timer will start at 0 when it is restarted. A tenth of a second and a hundredth of a second timers are available with a maximum time of 9999999.9 and 999999.99 seconds respectively. The timing diagram below shows the relationship between the timer input, timer reset, associated discrete bit, current value, and timer preset.



Timer (TMR) and Timer Fast (TMRF)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Timer instruction is a 0.1 second single-input timer that times to a maximum of 999.9 seconds. The Timer Fast instruction is a 0.01 second single input timer that times up to a maximum of 99.99 seconds. These timers will be enabled if the input logic is true (on) and will be reset to 0 if the input logic is false (off).

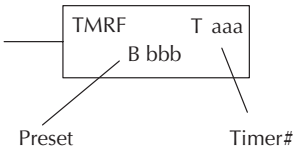


Instruction Specifications

DS	Used
HPP	Used

Timer Reference (Taaa): Specifies the timer number.

Preset Value (Bbbb): Constant value (K) or a V-memory location. (Pointer (P) for D2-240, D2-250–1, D2-260 and D2-262).



Current Value: Timer current values are accessed by referencing the associated V or T memory location. For example, the timer current value for T3 physically resides in V-memory location V3.

Discrete Status Bit: The discrete status bit is referenced by the associated T memory location. It will be ON if the current value is equal to or greater than the preset value. For example, the discrete status bit for Timer 2 would be T2.

NOTE: *A V-memory preset is required only if the ladder program or an Operator Interface unit must change the preset.*

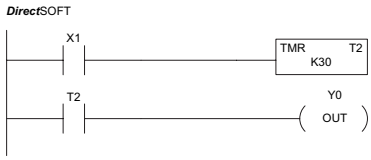
Operand Data Type	Range							
	D2-230		D2-240		D2-250-1		D2-260/D2-262	
B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
Timers T	0-77		0-177		0-377		0-377	
V-memory for preset values V	–	2000-2377	–	2000-3777	–	1400-7377 10000-17777	–	1400-7377 10000-37777
Pointers (presets only) P				2000-3777		1400-7377 10000-17777		1400-7377 10000-37777
Constants (presets only) K	–	0-9999	–	0-9999	–	0-9999	–	0-9999
Timer discrete status bits T/V*	0-77 or V41100-41103		0-177 or V41100-41107		0-377 or V41100-41117		0-377 or V41100-41117	
Timer current values V/T*	0-77		0-177		0-377		0-377	

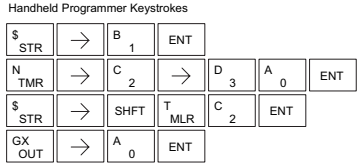
NOTE: **Both the Timer discrete status bits and the current value are accessed with the same data reference with the HPP.* ***Direct****SOFT uses separate references, such as "T2" for discrete status bit for Timer T2, and "TA2" for the current value of Timer T2.*

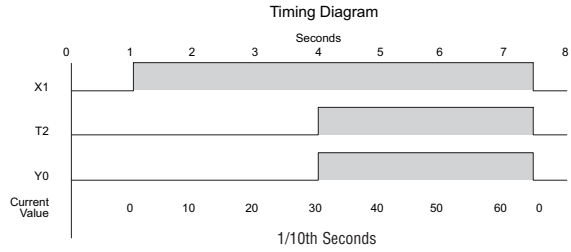
You can perform functions when the timer reaches the specified preset using the discrete status bit. Or, use the comparative contacts to perform functions at different time intervals based on one timer. The examples on the following page show these methods of programming timers.

Timer Example Using Discrete Status Bits

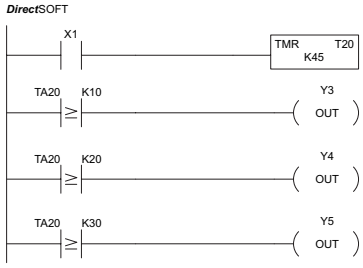
In the following example, a single-input timer is used with a preset of 3 seconds. The timer discrete status bit (T2) will turn on when the timer has timed for 3 seconds. The timer is reset when X1 turns off, turning the discrete status bit off and resetting the timer current value to 0.

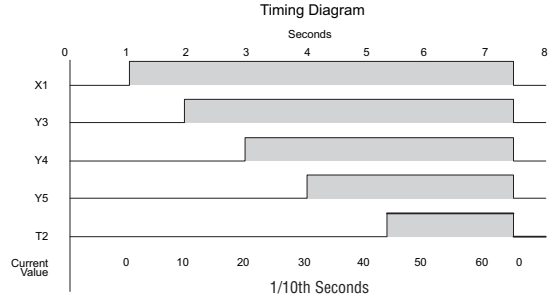


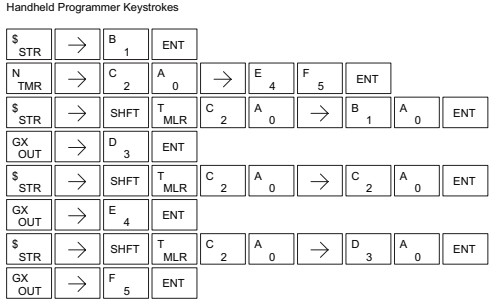




In the following example, a single-input timer is used with a preset of 4.5 seconds. Comparative contacts are used to energize Y3, Y4, and Y5 at one-second intervals respectively. When X1 is turned off, the timer will be reset to 0 and the comparative contacts will turn off Y3, Y4, and Y5.



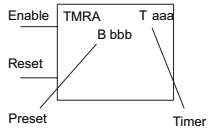




Accumulating Timer (TMRA)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

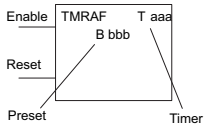
The Accumulating Timer is a 0.1 second two-input timer that will time to a maximum of 9999999.9. *The TMRA uses two timer registers in V-memory.*



Accumulating Fast Timer (TMRAF)

☑ 262

The Accumulating Fast Timer is a 0.01 second two-input timer that will time to a maximum of 999999.99. *The TMRAF uses two timer registers in V-memory.*



These timers have two inputs: an enable and a reset. The timer will start timing when the enable is on and stop timing when the enable is off without resetting the value to 0. The reset will reset the timer when on and allow the timer to time when off.

DS	Used
HPP	Used

Instruction Specifications

Timer Reference (Taaa): Specifies the timer number.

Preset Value (Bbbb): Constant value (K) or two consecutive V-memory locations. (Pointer (P) for D2-240, D2-250–1, D2-260 and D2-262).

Current Value: Timer current values are accessed by referencing the associated V or T memory location. For example, the timer current value for T3 resides in V-memory location V3.

Discrete Status Bit: The discrete status bit is accessed by referencing the associated T memory location. It will be on if the current value is equal to or greater than the preset value. For example, the discrete status bit for Timer 2 would be T2.

NOTE: *The accumulating timer uses two consecutive V-memory locations for the 8-digit value; therefore, two consecutive timer locations. For example, if TMRA T1 is used, the next available timer number is T3.*

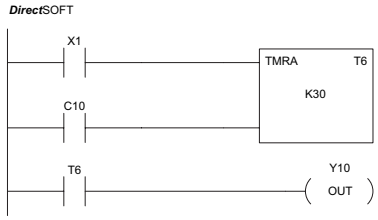
NOTE: *A V-memory preset is required only if the ladder program or an OIT must be used to change the preset.*

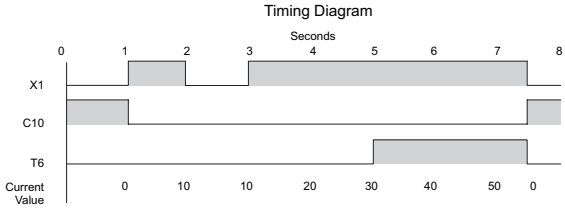
Operand Data Type		Range							
		D2-230		D2-240		D2-250-1		D2-260/D2-262	
	B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
Timers	T	0-77		0-177		0-377		0-377	
V-memory for preset values	V	–	2000-2377	–	2000-3777	–	1400-7377 10000-17777	–	1400-7377 10000-37777
Pointers (presets only)	P				2000-3777		1400-7377 10000-17777		1400-7377 10000-37777
Constants (presets only)	K	–	0-9999	–	0-9999	–	0-9999	–	0-9999
Timer discrete status bits	T/V*	0-77 or V41100-41103		0-177 or V41100-41107		0-377 or V41100-41117		0-377 or V41100-41117	
Timer current values	V/T*	0-77		0-177		0-377		0-377	

NOTE: ** Both the Timer discrete status bits and the current value are accessed with the same data reference with the HPP.* ***Direct****SOFT uses separate references, such as "T2" for discrete status bit for Timer T2, and "TA2" for the current value of Timer T2.*

Accumulating Timer Example using Discrete Status Bits

In the following example, a two input timer (accumulating timer) is used with a preset of three seconds. The timer discrete status bit (T6) will turn on when the timer has timed for three seconds. Notice in this example that the timer times for one second, stops for one second, then resumes timing. The timer will reset when C10 turns on, turning the discrete status bit off and resetting the timer current value to zero.





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
$ STR	→	SHFT	C 2	B 1	A 0	ENT		
N TMR	SHFT	A 0	→	G 6	→			

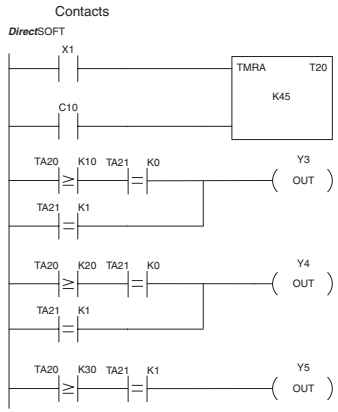
1/10th Seconds

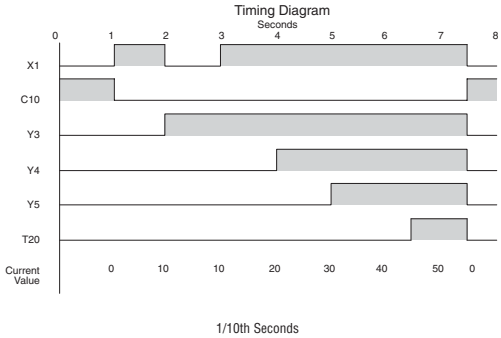
Handheld Programmer Keystrokes (cont'd)

D 3	A 0	ENT						
$ STR	→	SHFT	T MLR	G 6	ENT			
GX OUT	→	B 1	A 0	ENT				

Accumulator Timer Example Using Comparative Contacts

In the following example, a two-input timer is used with a preset of 4.5 seconds. Comparative contacts are used to energize Y3, Y4, and Y5 at one-second intervals respectively. The comparative contacts will turn off when the timer is reset.





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT									
$ STR	→	SHFT	C 2	B 1	A 0	ENT						
N TMR	SHFT	A 0	→	C 2	A 0	→	E 4	F 5	ENT			
$ STR	→	SHFT	T MLR	C 2	A 0	→	B 1	A 0	ENT			
V AND	SHFT	E 4	→	SHFT	T MLR	C 2	B 1	→	A 0	ENT		
Q OR	SHFT	E 4	→	SHFT	T MLR	C 2	B 1	→	B 1	ENT		
GX OUT	→	D 3	ENT									

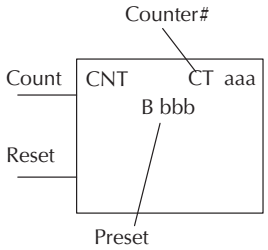
Handheld Programmer Keystrokes (cont'd)

$ STR	→	SHFT	T MLR	C 2	A 0	→	C 2	A 0	ENT			
V AND	SHFT	E 4	→	SHFT	T MLR	C 2	B 1	→	A 0	ENT		
Q OR	SHFT	E 4	→	SHFT	T MLR	C 2	B 1	→	B 1	ENT		
GX OUT	→	E 4	ENT									
$ STR	→	SHFT	T MLR	C 2	A 0	→	D 3	A 0	ENT			
V AND	SHFT	E 4	→	SHFT	T MLR	C 2	B 1	→	B 1	ENT		
GX OUT	→	F 5	ENT									

Counter (CNT)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Counter is a two-input counter that increments wl count input logic transitions from off to on. When the reset input is on, the counter resets to zero. When the value equals the preset value, the counter status bit co and the counter continues to count up to a maximun of 9999. The maximum value will be held until the is reset.



DS	Used
HPP	Used

Instruction Specifications

Counter Reference (CTaaa): Specifies the counter number.

Preset Value (Bbbb): Constant value (K) or a V-memory location. (Pointer (P) for D2-240, D2-250–1, D2-260 and D2-262.)

Current Values: Counter current values are accessed by referencing the associated V or CT memory locations. The V-memory location is the counter location + 1000. For example, the counter current value for CT3 resides in V-memory location V1003.

Discrete Status Bit: The discrete status bit is accessed by referencing the associated CT memory location. It will be on if the value is equal to or greater than the preset value. For example, the discrete status bit for Counter 2 would be CT2.

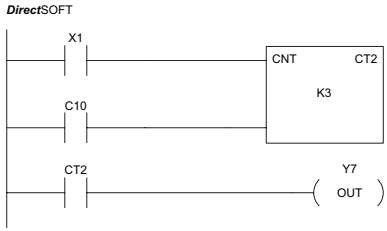
NOTE: *A V-memory preset is required only if the ladder program or an OIT must used to change the preset*

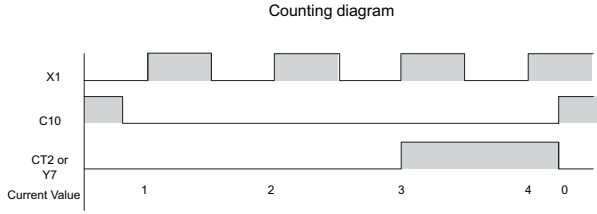
Operand Data Type	Range							
	D2-230		D2-240		D2-250-1		D2-260/D2-262	
B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
Counters CT	0-77		0-177		0-177		0-377	
V-memory for preset values V	–	2000-2377	–	2000-3777	–	1400-7377 10000-17777	–	1400-7377 10000-37777
Pointers (presets only) P				2000-3777		1400-7377 10000-17777		1400-7377 10000-37777
Constants (presets only) K	–	0-9999	–	0-9999	–	0-9999	–	0-9999
Counter discrete status bits CT/V*	0-77 or V41140-41143		0-177 or V41140-41147		0-177 or V41140-41147		0-377 or V41100-41157	
Counter current values V/CT*	1000-1077		1000-1177		1000-1177		1000-1377	

NOTE: ** Both the Counter discrete status bits and the current value are accessed with the same data reference with the HPP.* ***Direct****SOFT uses separate references, such as "CT2" for discrete status bit for Counter CT2, and "CTA2" for the current value of Counter CT2.*

Counter Example Using Discrete Status Bits

In the following example, when X1 makes an off-to-on transition, counter CT2 will increment by one. When the current value reaches the preset value of 3, the counter status bit CT2 will turn on and energize Y7. When the reset C10 turns on, the counter status bit will turn off and the current value will be 0. The current value for counter CT2 will be held in V-memory location V1002.





Handheld Programmer Keystrokes

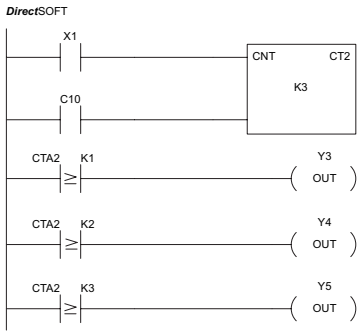
$ STR	→	B 1	ENT			
$ STR	→	SHFT	C 2	B 1	A 0	ENT
GY CNT	→	C 2	→	D 3	ENT	

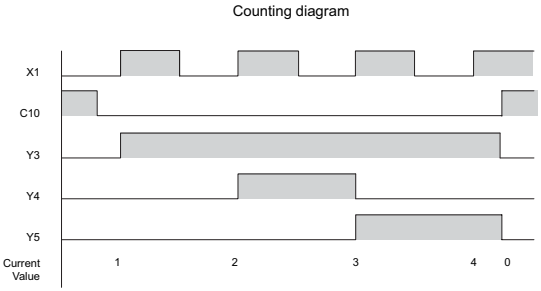
Handheld Programmer Keystrokes (cont)

$ STR	→	SHFT	C 2	SHFT	T MLR	C 2	ENT
GX OUT	→	H 7	ENT				

Counter Example Using Comparative Contacts

In the following example, when X1 makes an off-to-on transition, counter CT2 will increment by one. Comparative contacts are used to energize Y3, Y4, and Y5 at different counts. When the reset C10 turns on, the counter status bit will turn off and the counter current value will be 0, and the comparative contacts will turn off.





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT				
$ STR	→	SHFT	C 2	B 1	A 0	ENT	
GY CNT	→	C 2	→	D 3	ENT		
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2	
→	B 1	ENT					
GX OUT	→	D 3	ENT				

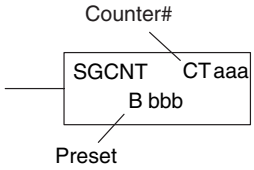
Handheld Programmer Keystrokes (cont)

$ STR	→	SHFT	C 2	SHFT	T MLR	C 2
→	C 2	ENT				
GX OUT	→	E 4	ENT			
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2
→	D 3	ENT				
GX OUT	→	F 5	ENT			

Stage Counter (SGCNT)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Stage Counter is a single-input counter that increments when the input logic transitions from off to on. This counter differs from other counters since it will hold its current value until reset using the RST instruction. The Stage Counter is designed for use in RLL*PLUS* programs but can be used in relay ladder logic programs. When the current value equals the preset value, the counter status bit turns on and the counter continues to count up to a maximum count of 9999. The maximum value will be held until the counter is reset.



The counter discrete status bit and the current value are not specified in the counter instruction.

DS	Used
HPP	Used

Instruction Specifications

Counter Reference (CTaaa): Specifies the counter number.

Preset Value (Bbbb): Constant value (K) or a V-memory location.(Pointer (P) for D2-240, D2-250–1, D2-260 and D2-262.)

Current Values: Counter current values are accessed by referencing the associated V or CT memory locations. The V-memory location is the counter location + 1000. For example, the counter current value for CT3 resides in V-memory location V1003.

Discrete Status Bit: The discrete status bit is accessed by referencing the associated CT memory location. It will be on if the value is equal to or greater than the preset value. For example, the discrete status bit for Counter 2 would be CT2.

NOTE: *When using a counter inside a stage, the stage must be active for one scan before the input to the counter makes a 0-1 transition. Otherwise, there is no real transition and the counter will not count.*

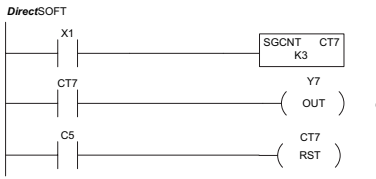
NOTE: *A V-memory preset is required only if the ladder program or an OIT must used to change the preset.*

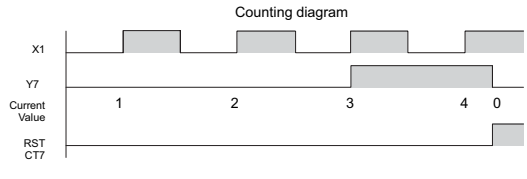
Operand Data Type	Range							
	D2-230		D2-240		D2-250-1		D2-260/D2-262	
B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
Counters CT	0-77		0-177		0-177		0-377	
V-memory for preset values V	–	2000-2377	–	2000-3777	–	1400-7377 10000-17777	–	1400-7377 10000-37777
Pointers (presets only) P				2000-3777		1400-7377 10000-17777		1400-7377 10000-37777
Constants (presets only) K	–	0-9999	–	0-9999	–	0-9999	–	0-9999
Counter discrete status bits CT/V*	0-77 or V41140-41143		0-177 or V41140-41147		0-177 or V41140-41147		0-377 or V41100-41157	
Counter current values V/CT*	1000-1077		1000-1177		1000-1177		1000-1377	

NOTE: ** Both the Counter discrete status bits and the current value are accessed with the same data reference with the HPP.* ***Direct****SOFT uses separate references, such as "CT2" for discrete status bit for Counter CT2, and "CTA2" for the current value of Counter CT2.*

Stage Counter Example Using Discrete Status Bits

In the following example, when X1 makes an off-to-on transition, stage counter CT7 will increment by one. When the current value reaches 3, the counter status bit CT7 will turn on and energize Y7. The counter status bit CT7 will remain on until the counter is reset using the RST instruction. When the counter is reset, the counter status bit will turn off and the counter current value will be 0. The current value for counter CT7 will be held in V-memory location V1007.





Handheld Programmer Keystrokes

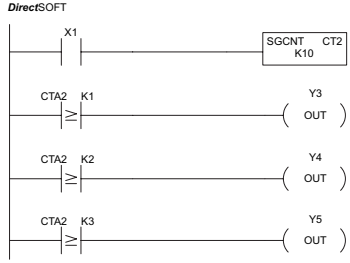
$ STR	→	B 1	ENT										
SHFT	S RST	SHFT	G 6	SHFT	GY CNT	→							
H 7	→	D 3	ENT										
$ STR	→	SHFT	C 2	SHFT	T MLR	H 7	ENT						

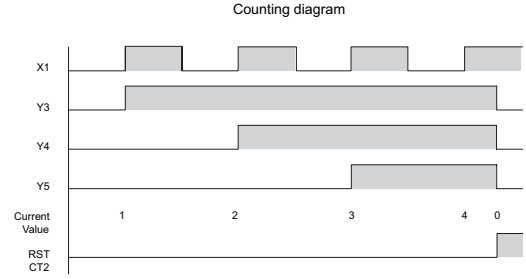
Handheld Programmer Keystrokes (cont)

GX OUT	→	H 7	ENT										
$ STR	→	SHFT	C 2	F 5	ENT								
S RST	→	SHFT	C 2	SHFT	T MLR	H 7	ENT						

Stage Counter Example Using Comparative Contacts

In the following example, when X1 makes an off-to-on transition, counter CT2 will increment by one. Comparative contacts are used to energize Y3, Y4, and Y5 at different counts. Although this is not shown in the example, when the counter is reset using the Reset instruction, the counter status bit will turn off and the current value will be 0. The current value for counter CT2 will be held in V-memory location V1002.





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT										
SHFT	S RST	G 6	SHFT	GY CNT	→								
C 2	→	B 1	A 0	ENT									
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2							
→	B 1	ENT											
GX OUT	→	D 3	ENT										

Handheld Programmer Keystrokes (cont)

$ STR	→	SHFT	C 2	SHFT	T MLR	C 2							
→	C 2	ENT											
GX OUT	→	E 4	ENT										
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2							
→	D 3	ENT											
GX OUT	→	F 5	ENT										

Up Down Counter (UDC)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

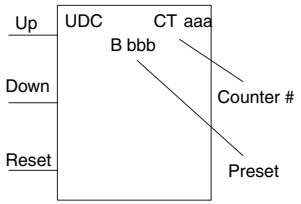
This Up/Down Counter counts up on each off-to-on transition of the Up input and counts down on each off to on transition of the Down input. The counter is reset to 0 when the Reset input is on. The count range is 0 to 99999999. The count input not being used must be off in order for the active count input to function.

DS	Used
HPP	Used

Instruction Specification

Counter Reference (CTaaa): Specifies the counter number.

Preset Value (Bbbb): Constant value (K) or two consecutive V-memory locations. (Pointer (P) for D2-240, D2-250–1, D2-260 and D2-262).



Caution: The UDC uses two V-memory locations for the 8-digit current value. This means that the UDC uses two consecutive counter locations. If UDC CT1 is used in the program, the next available counter is CT3.

Current Values: Current count is a double word value accessed by referencing the associated V or CT memory locations. The V-memory location is the counter location + 1000. For example, the counter current value for CT5 resides in V-memory location V1005 and V1006.

Discrete Status Bit: The discrete status bit is accessed by referencing the associated CT memory location. It will be on if the value is equal to or greater than the preset value. For example, the discrete status bit for Counter 2 would be CT2.

NOTE: *The UDC uses two consecutive V-memory locations for the 8-digit value, therefore two consecutive counter locations. For example, if UDC CT1 is used, the next available counter number is CT3.*

NOTE: *A V-memory preset is required only if the ladder program or an OIT must be used to change the preset.*

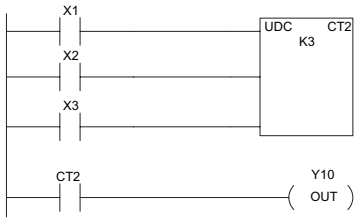
Operand Data Type	Range							
	D2-230		D2-240		D2-250-1		D2-260/D2-262	
B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
Counters CT	0-76		0-176		0-176		0-376	
V-memory for preset values V	–	2000-2377	–	2000-3777	–	1400-7377 10000-17777	–	1400-7377 10000-37777
Pointers (presets only) P				2000-3777		1400-7377 10000-17777		1400-7377 10000-37777
Constants (presets only) K	–	0-99999999	–	0-99999999	–	0-99999999	–	0-99999999
Counter discrete status bits CT/V*	0-76 or V41140-41143		0-176 or V41140-41147		0-176 or V41140-41147		0-376 or V41100-41157	
Counter current values V/CT*	1000-1076		1000-1176		1000-1176		1000-1376	

NOTE: ** Both the Counter discrete status bits and the current value are accessed with the same data reference with the HPP.* ***Direct****SOFT uses separate references, such as "CT2" for discrete status bit for Counter CT2, and "CTA2" for the current value of Counter CT2.*

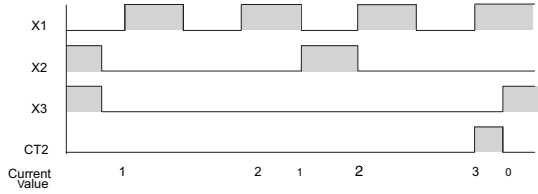
Up/Down Counter Example Using Discrete Status Bits

In the following example, if X2 and X3 are off, when X1 toggles from off to on the counter will increment by one. If X1 and X3 are off, the counter will decrement by one when X2 toggles from off to on. When the count value reaches the preset value of 3, the counter status bit will turn on. When the reset X3 turns on, the counter status bit will turn off and the current value will be 0.

*Direct*SOFT



Counting Diagram



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT
$ STR	→	C 2	ENT
$ STR	→	D 3	ENT
SHFT	U ISG	D 3	C 2	→	C 2

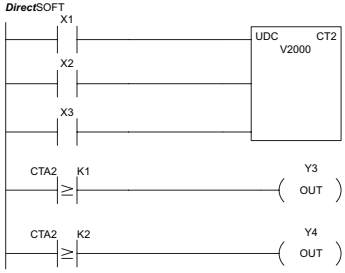
Handheld Programmer Keystrokes (cont)

→	D 3	ENT					
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2	ENT
GX OUT	→	B 1	A 0	ENT			

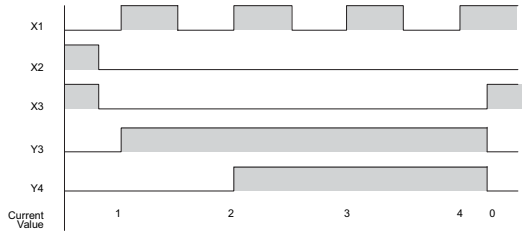
Up/Down Counter Example Using Comparative Contacts

In the following example, when X1 makes an off to on transition, counter CT2 will increment by one. Comparative contacts are used to energize Y3 and Y4 at different counts. When the reset (X3) turns on, the counter status bit will turn off, the current value will be 0, and the comparative contacts will turn off.

***Direct*SOFT**



Counting Diagram



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT				
$ STR	→	C 2	ENT				
$ STR	→	D 3	ENT				
SHFT	U ISG	D 3	C 2	→	C 2	→	
SHFT	V AND	C 2	A 0	A 0	A 0	ENT	
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2	

Handheld Programmer Keystrokes (cont)

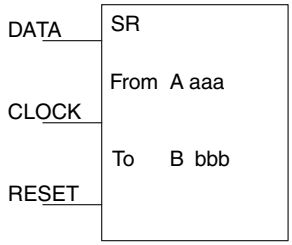
→	B 1	ENT				
GX OUT	→	D 3	ENT			
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2
→	C 2	ENT				
GX OUT	→	E 4	ENT			

Shift Register (SR)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Shift Register instruction shifts data through a predefined number of control relays. The control ranges in the shift register block must start at the beginning of an 8-bit boundary and use 8-bit blocks.

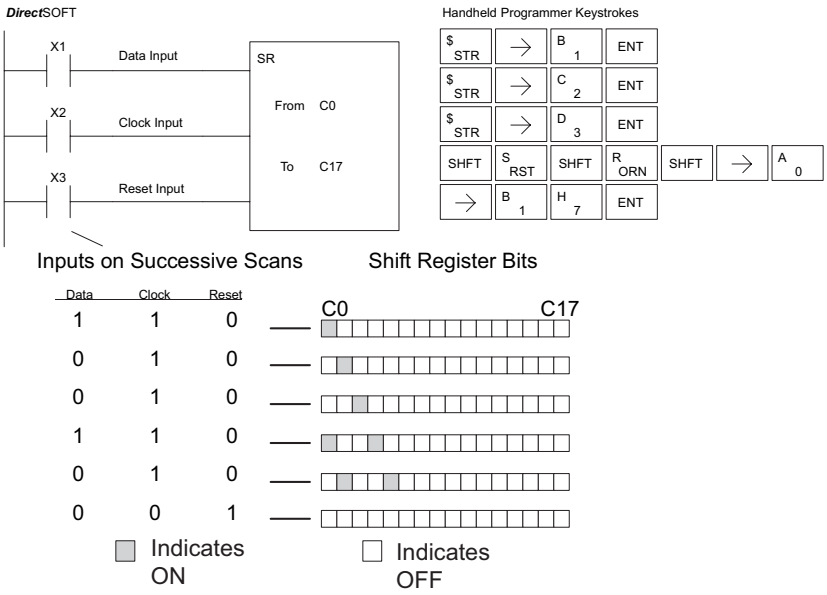


The Shift Register has three contacts.

- Data — determines the value (1 or 0) that will enter the register
- Clock — shifts the bits one position on each low to high transition
- Reset —resets the Shift Register to all zeros.

With each off-to-on transition of the clock input, the bits which make up the shift register block are shifted by one bit position and the status of the data input is placed into the starting bit position in the shift register. The direction of the shift depends on the entry in the From and To fields. From C0 to C17 would define a block of 16 bits to be shifted from left to right. From C17 to C0 would define a block of 16 bits, to be shifted from right to left. The maximum size of the shift register block depends on the number of available control relays. The minimum block size is 8 control relays.

Operand Data Type		Range							
		D2-230		D2-240		D2-250-1		D2-260/D2-262	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb	aaa	bbb
Control Relay	C	0-377	0-377	0-377	0-377	0-1777	0-1777	0-3777	0-3777



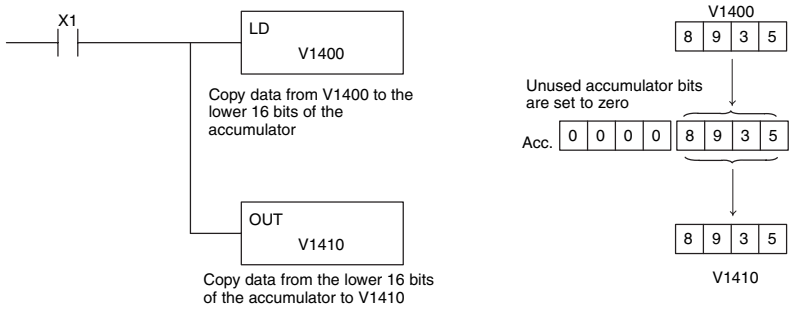
Accumulator/Stack Load and Output Data Instructions

Using the Accumulator

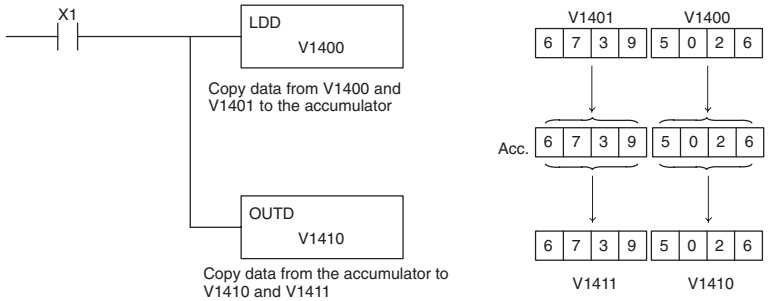
The accumulator in the DL205 series CPUs is a 32-bit register that is used as a temporary storage location for data that is being copied or manipulated in some manner. For example, you have to use the accumulator to perform math operations, such as, add, subtract, multiply, etc. Since there are 32 bits, you can use up to an 8-digit BCD number or a 32-bit 2's compliment number. The accumulator is reset to 0 at the end of every CPU scan.

Copying Data to the Accumulator

The Load and Out instructions and their variations are used to copy data from a V-memory location to the accumulator, or to copy data from the accumulator to V-memory. The following example copies data from V-memory location V1400 to V-memory location V1410.

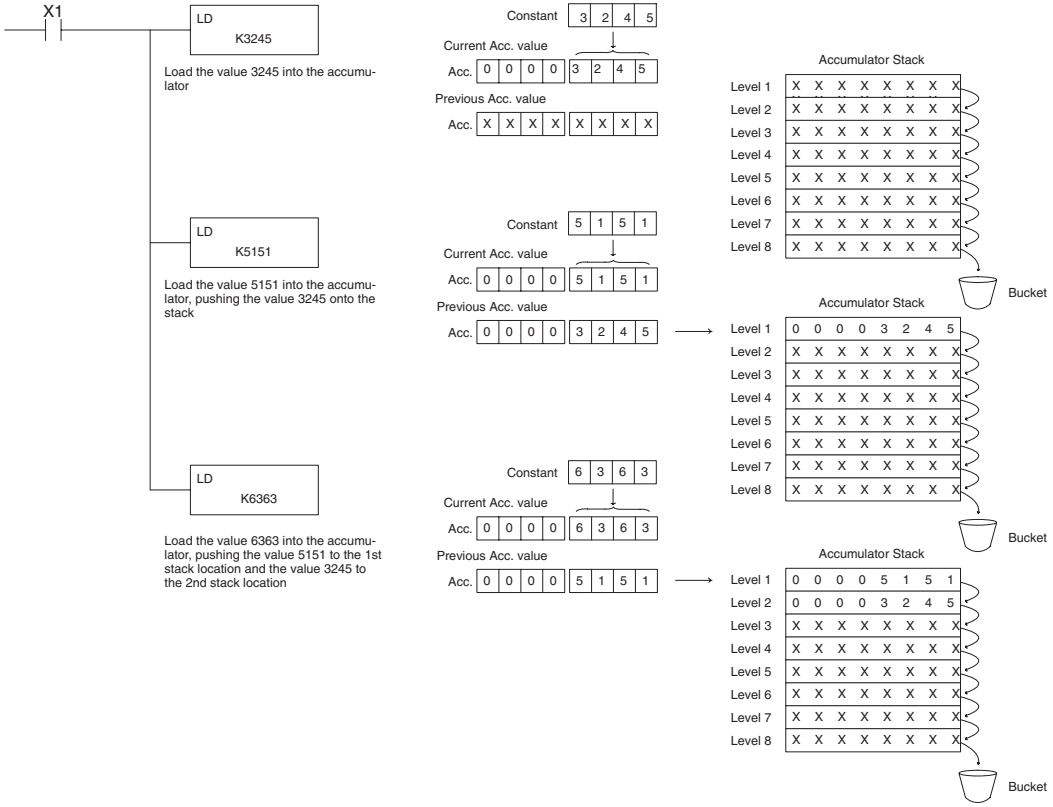


Since the accumulator is 32 bits and V-memory locations are 16 bits, the Load Double and Out Double (or variations thereof) use two consecutive V-memory locations or 8-digit BCD constants to copy data either to the accumulator from a V-memory address or from a V-memory address to the accumulator. For example, if you wanted to copy data from V1400 and V1401 to V1410 and V1411, the most efficient way to perform this function would be as follows:

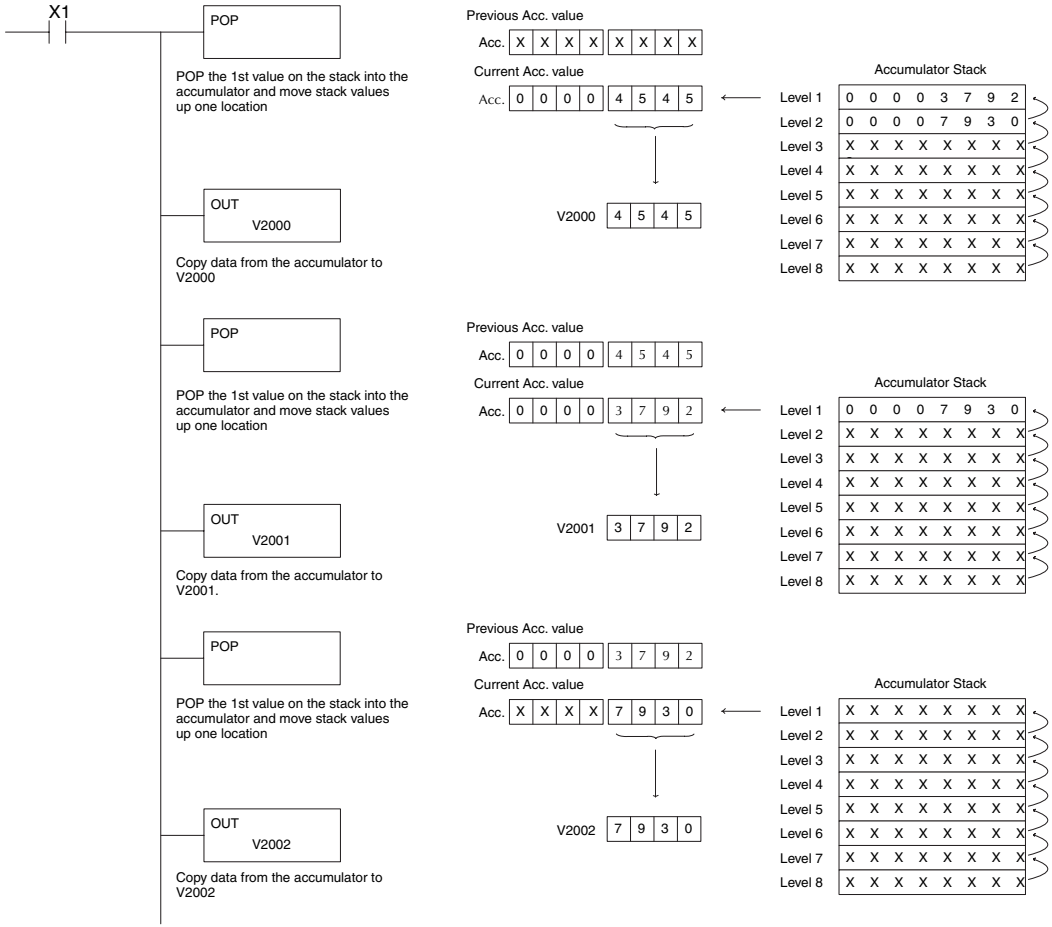


Using the Accumulator Stack

The accumulator stack is used for instructions that require more than one parameter to execute a function or for user-defined functionality. The accumulator stack is used when more than one Load instruction is executed without the use of an Out instruction. The first Load instruction in the scan places a value into the accumulator. Every Load instruction thereafter without the use of an Out instruction places a value into the accumulator and the value that was in the accumulator is placed onto the accumulator stack. The Out instruction nullifies the previous Load instruction and does not place the value that was in the accumulator onto the accumulator stack when the next Load instruction is executed. Every time a value is placed onto the accumulator stack, the other values in the stack are pushed down one location. The accumulator is eight levels deep (eight 32-bit registers). If there is a value in the eighth location when a new value is placed onto the stack, the value in the eighth location is pushed off the stack and cannot be recovered.

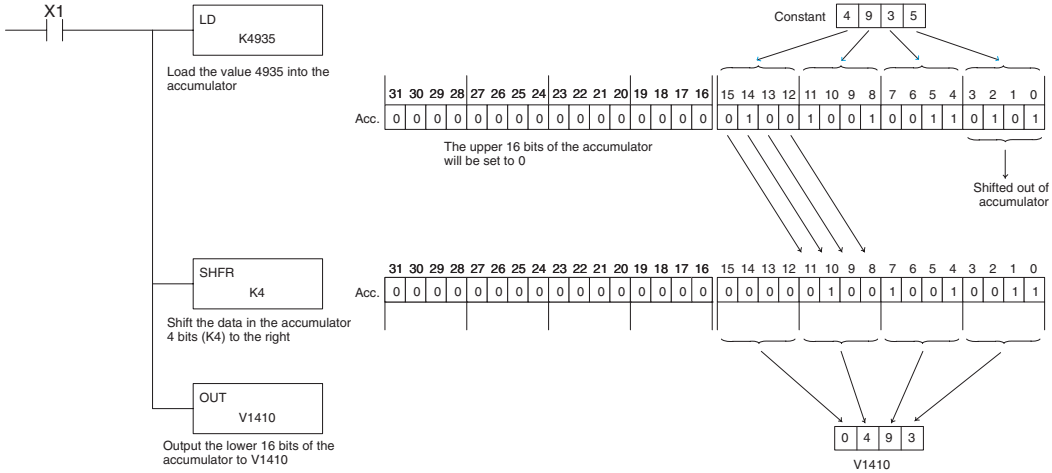


The POP instruction rotates values upward through the stack into the accumulator. When a POP is executed, the value that was in the accumulator is cleared and the value that was on top of the stack is in the accumulator. The values in the stack are shifted up one position in the stack.


X1
POP
POP the 1st value on the stack into the accumulator and move stack values up one location
OUT
V2000
Copy data from the accumulator to V2000
POP
POP the 1st value on the stack into the accumulator and move stack values up one location
OUT
V2001
Copy data from the accumulator to V2001.
POP
POP the 1st value on the stack into the accumulator and move stack values up one location
OUT
V2002
Copy data from the accumulator to V2002
Previous Acc. value
Acc. X X X X X X X X
Current Acc. value
Acc. 0 0 0 0 4 5 4 5
V2000 4 5 4 5
Accumulator Stack
Level 1 0 0 0 0 3 7 9 2
Level 2 0 0 0 0 7 9 3 0
Level 3 X X X X X X X X
Level 4 X X X X X X X X
Level 5 X X X X X X X X
Level 6 X X X X X X X X
Level 7 X X X X X X X X
Level 8 X X X X X X X X
Previous Acc. value
Acc. 0 0 0 0 4 5 4 5
Current Acc. value
Acc. 0 0 0 0 3 7 9 2
V2001 3 7 9 2
Accumulator Stack
Level 1 0 0 0 0 7 9 3 0
Level 2 X X X X X X X X
Level 3 X X X X X X X X
Level 4 X X X X X X X X
Level 5 X X X X X X X X
Level 6 X X X X X X X X
Level 7 X X X X X X X X
Level 8 X X X X X X X X
Previous Acc. value
Acc. 0 0 0 0 3 7 9 2
Current Acc. value
Acc. X X X X 7 9 3 0
V2002 7 9 3 0
Accumulator Stack
Level 1 X X X X X X X X
Level 2 X X X X X X X X
Level 3 X X X X X X X X
Level 4 X X X X X X X X
Level 5 X X X X X X X X
Level 6 X X X X X X X X
Level 7 X X X X X X X X
Level 8 X X X X X X X X

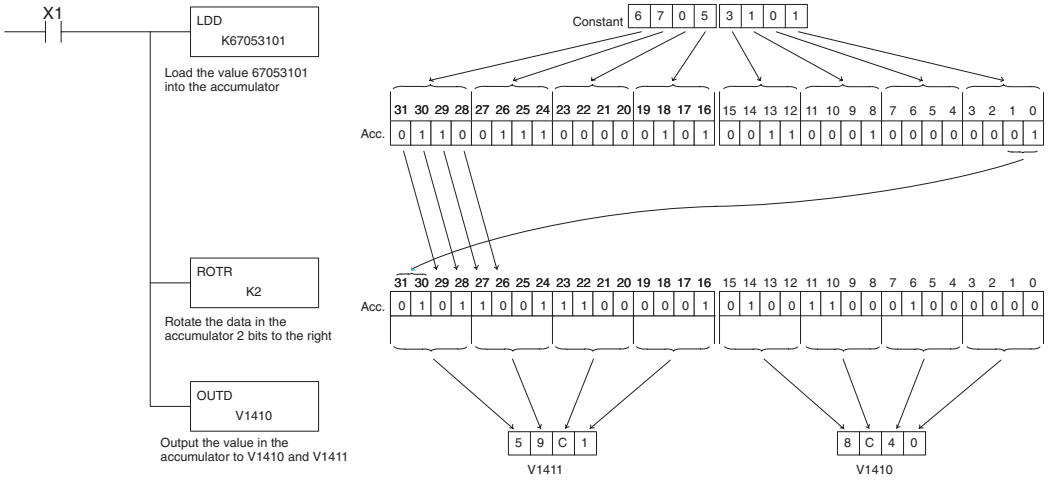
Changing the Accumulator Data

Instructions that manipulate data also use the accumulator. The result of the manipulated data resides in the accumulator. The data that was being manipulated is cleared from the accumulator. The following example loads the constant value 4935 into the accumulator, shifts the data right 4 bits, and outputs the result to V1410.



Some of the data manipulation instructions use 32 bits. They use two consecutive V-memory locations or an 8-digit BCD constant to manipulate data in the accumulator.

The following example rotates the value 67053101 two bits to the right and outputs the value to V1410 and V1411.

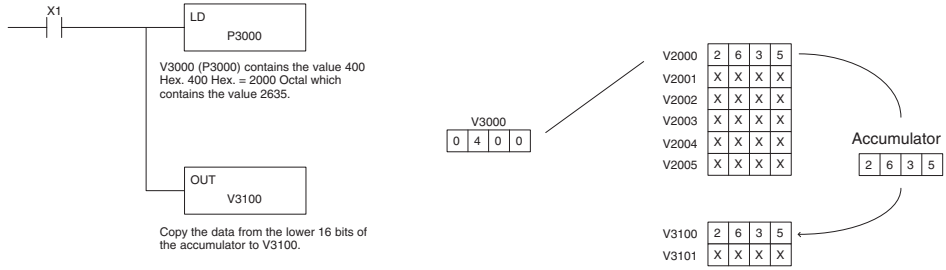


Using Pointers

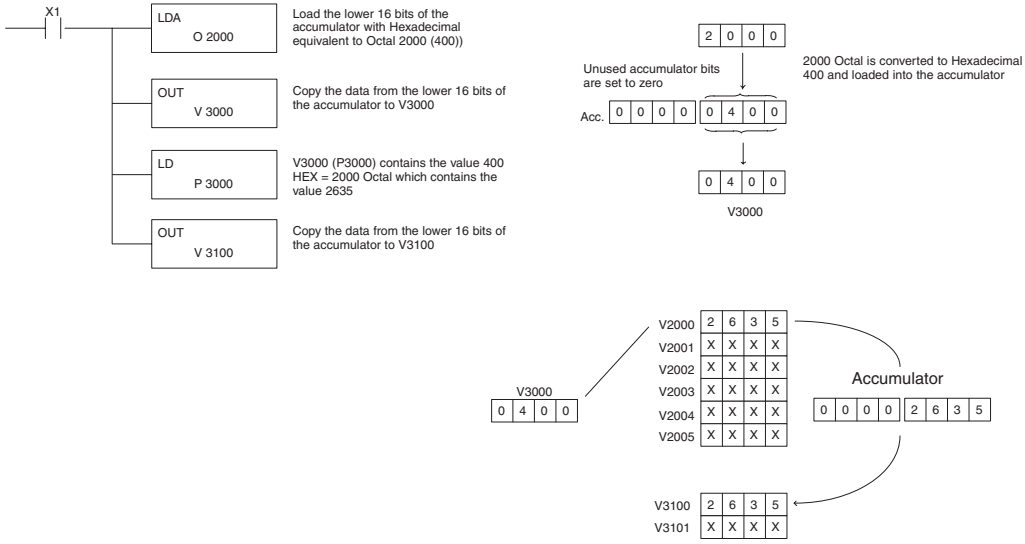
Many of the DL205 series instructions will allow V-memory pointers as an operand. Pointers can be useful in ladder logic programming, but can be difficult to understand or implement in your application if you do not have prior experience with pointers (commonly known as indirect addressing). Pointers allow instructions to obtain data from V-memory locations referenced by the pointer value.

NOTE: *In the DL205, V-memory addressing is in octal. However the value in the pointer location which will reference a V-memory location is viewed as HEX. Use the Load Address instruction to move an address into the pointer location. This instruction performs the Octal to Hexadecimal conversion for you.*

The following example uses a pointer operand in a Load instruction. V-memory location 3000 is the pointer location. V3000 contains the value 400, which is the HEX equivalent of the Octal address V-memory location V2000. The CPU copies the data from V2000 into the lower word of the accumulator.



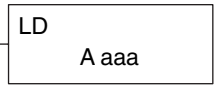
The following example is similar to the one above, except for the LDA (load address) instruction, which automatically converts the Octal address to the Hex equivalent.



Load (LD)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Load instruction is a 16-bit instruction that loads the value (Aaaa), which is either a V-memory location or a 4-digit constant, into the lower 16 bits of the accumulator. The upper 16 bits of the accumulator are set to 0.



Operand Data Type		Range				
		D2-230	D2-240	D2-250-1	D2-260/D2-262	
	A	aaa	aaa	aaa	aaa	bbb
V-memory	V	All See Memory map	All, see Memory map			
Pointer	P	–	All V-memory; see Memory map			
Constant	K	0-FFFF	0-FFFF			

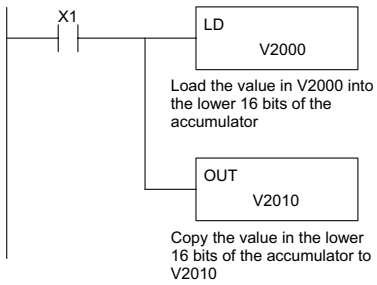
Discrete Bit Flags	Description
SP76	On when the value loaded into the accumulator by any instruction is zero.

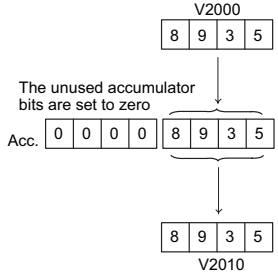
NOTE: *Two consecutive Load instructions will place the value of the first Load instruction onto the accumulator stack.*

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator and output to V2010.

DS	Used
HPP	Used

DirectSOFT





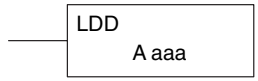
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→					
C 2	A 0	A 0	A 0	ENT				
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

Load Double (LDD)

The Load Double instruction is a 32-bit instruction that loads the value (Aaaa), which is either two consecutive V-memory locations or an 8-digit constant value, into the accumulator.



Operand Data Type		Range				
		D2-230	D2-240	D2-250-1	D2-260/D2-262	
	A	aaa	aaa	aaa	aaa	bbb
V-memory	V	All; see Memory map				
Pointer	P	–	All V-memory; see Memory map			
Constant	K	0-FFFFFFFF	0-FFFFFFFF			

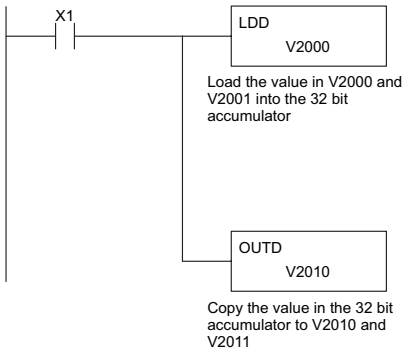
Discrete Bit Flags	Description
SP76	On when the value loaded into the accumulator by any instruction is zero.

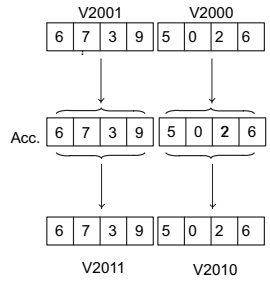
NOTE: *Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.*

In the following example, when X1 is on, the 32-bit value in V2000 and V2001 will be loaded into the accumulator and output to V2010 and V2011.

DS	Used
HPP	Used

DirectSOFT





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT	
SHFT	L ANDST	D 3	D 3	→
C 2	A 0	A 0	A 0	ENT
GX OUT	SHFT	D 3	→	
C 2	A 0	B 1	A 0	ENT

Load Formatted (LDF)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Load Formatted instruction loads 1 to 32 consecutive bits from discrete memory locations into the accumulator. The instruction requires a starting location (Aaaa) and the number of bits (Kbbb) to be loaded. Unused accumulator bit locations are set to zero.

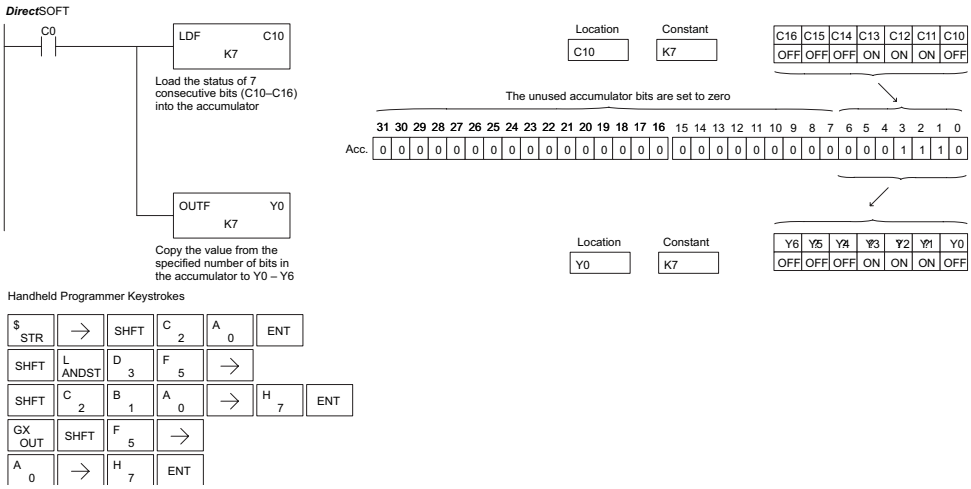
Operand Data Type		Range					
		D2-240		D2-250-1		D2-260/D2-262	
	A	aaa	bbb	aaa	bbb	aaa	bbb
Inputs	X	0–477	–	0–777	–	0–1777	–
Outputs	Y	0–477	–	0–777	–	0–1777	–
Control Relays	C	0–377	–	0–1777	–	0–3777	–
Stage bits	S	0–777	–	0–1777	–	0–1777	–
Timer bits	T	0–177	–	0–377	–	0–377	–
Counter bits	CT	0–177	–	0–177	–	0–377	–
Special Relays	SP	0-137, 540-617	–	0–777	–	0–777	–
Global I/O	GX/GY	–	–	–	–	0–3777	–
Constant	K	–	1–32	–	1–32	–	1–32

Discrete Bit Flags	Description
SP76	ON when the value loaded into the accumulator by any instruction is zero.

NOTE: *Two consecutive Load instructions will place the value of the first Load instruction onto the accumulator stack.*

DS	Used
HPP	Used

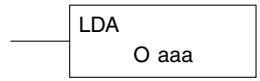
In the following example, when C0 is on, the binary pattern of C10–C16 (7 bits) will be loaded into the accumulator using the Load Formatted instruction. The lower 7 bits of the accumulator are output to Y0–Y6 using the Out Formatted instruction.



Load Address (LDA)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Load Address instruction is a 16-bit instruction. It converts any octal value or address to the HEX equivalent value and loads the HEX value into the accumulator. This instruction is useful when an address parameter is required since all addresses for the DL205 system are in octal.



Operand Data Type	Range			
	D2-230	D2-240	D2-250-1	D2-260/D2-262
	aaa	aaa	aaa	aaa
Octal Address O	All V-memory; see Memory map			

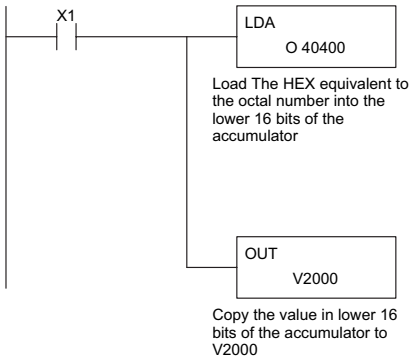
Discrete Bit Flags	Description
SP76	ON when the value loaded into the accumulator by any instruction is zero.

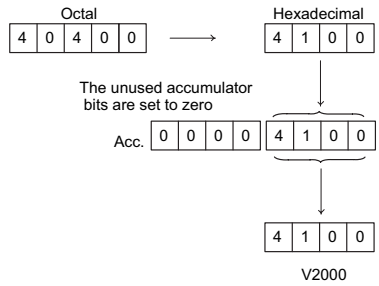
NOTE: *Two consecutive Load instructions will place the value of the first Load instruction onto the accumulator stack.*

In the following example, when X1 is on, the octal number 40400 will be converted to a HEX 4100 and loaded into the accumulator using the Load Address instruction. The value in the lower 16 bits of the accumulator is copied to V2000 using the Out instruction.

DS	Used
HPP	Used

DirectSOFT





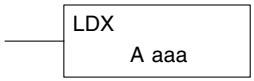
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	A 0	→				
E 4	A 0	E 4	A 0	A 0	ENT			
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	A 0	ENT

Load Accumulator Indexed (LDX)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

Load Accumulator Indexed is a 16-bit instruction that specifies a source address (V-memory) which will be offset by the value in the first stack location. This instruction interprets the value in the first stack location as HEX. The value in the offset address (source address + offset) is loaded into the lower 16 bits of the accumulator. The upper 16 bits of the accumulator are set to 0.



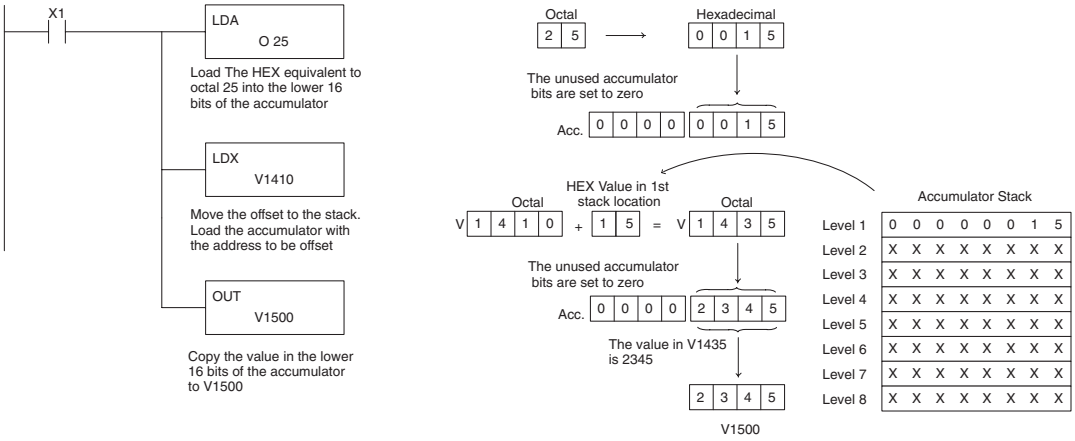
DS	Used
HPP	Used

Helpful hint: — The Load Address instruction can be used to convert an octal address to a HEX address and load the value into the accumulator.

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All. See memory map	All. See memory map
Pointer	P	All V-memory. See memory map	All V-memory. See memory map

NOTE: *Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.*

In the following example, when X1 is on, the HEX equivalent for octal 25 will be loaded into the accumulator (this value will be placed on the stack when the Load Accumulator Indexed instruction is executed). V-memory location V1410 will be added to the value in the first level of the stack and the value in this location (V1435 = 2345) is loaded into the lower 16 bits of the accumulator using the Load Accumulator Indexed instruction. The value in the lower 16 bits of the accumulator is output to V1500 using the Out instruction.

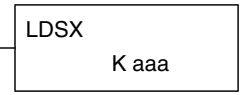


Load Accumulator Indexed from Data Constants (LDSX)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Load Accumulator Indexed from Data Constants is a 16-bit instruction. The instruction specifies a Data Label Area (DLBL) where numerical or ASCII constants are stored. This value will be loaded into the lower 16 bits.



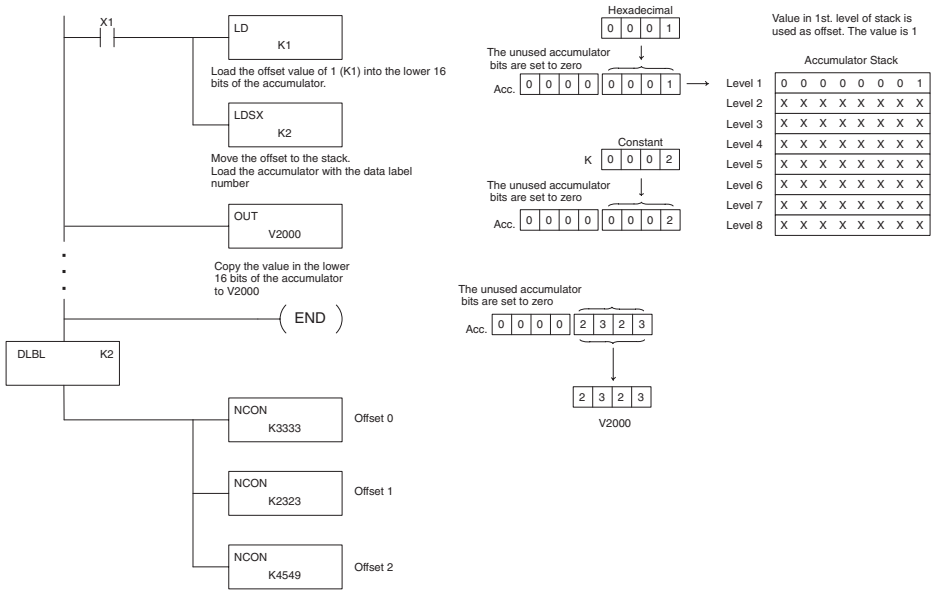
The LDSX instruction uses the value in the first level of the accumulator stack as an offset to determine which numerical or ASCII constant within the Data Label Area will be loaded into the accumulator. The LDSX instruction interprets the value in the first level of the accumulator stack as a HEX value.

Helpful hint: — The Load Address instruction can be used to convert octal to HEX and load the value into the accumulator.

Operand Data Type		Range		
		D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa
Constant	K	1-FFFF		

NOTE: *Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.*

In the following example, when X1 is on, the offset of 1 is loaded into the accumulator. This value will be placed into the first level of the accumulator stack when the LDSX instruction is executed. The LDSX instruction specifies the Data Label (DLBL K2) where the numerical constant(s) are located in the program and loads the constant value, indicated by the offset in the stack, into the lower 16 bits of the accumulator.



Handheld Programmer Keystrokes

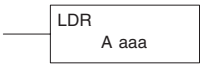
$ STR	→	B 1	ENT							
SHFT	L ANDST	D 3	→	SHFT	K JMP	B 1	ENT			
SHFT	L ANDST	D 3	S RST	X SET	→	C 2	ENT			
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	A 0	ENT		
SHFT	E 4	N TMR	D 3	ENT						
SHFT	D 3	L ANDST	B 1	L ANDST	→	C 2	ENT			
SHFT	N TMR	C 2	O INST#	N TMR	→	D 3	D 3	D 3	D 3	ENT
SHFT	N TMR	C 2	O INST#	N TMR	→	C 2	D 3	C 2	D 3	ENT
SHFT	N TMR	C 2	O INST#	N TMR	→	E 4	F 5	E 4	J 9	ENT

Load Real Number (LDR)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

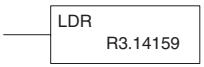
DS	Used
HPP	N/A

The Load Real Number instruction loads a real number contained in two consecutive V-memory locations, or an 8-digit constant into the accumulator.

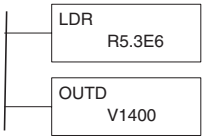


Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All. See memory map	All. See memory map
Pointer	P	All V-memory. See memory map	All V-memory. See memory map
Real Constants	R	-3.402823E+038 to +3.402823E+038	-3.402823E+038 to +3.402823E+038

*Direct*SOFT allows you to enter real numbers directly, by using the leading "**R**" to indicate a *real number* entry. You can enter a constant such as Pi, shown in the example to the right. To enter negative numbers, use a minus (–) after the "R".

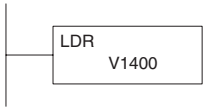


For very large numbers or very small numbers, you can use exponential notation. The number to the right is 5.3 million. The OUTD instruction stores it in V1400 and V1401.



These real numbers are in the IEEE 32-bit floating point format, so they occupy two V-memory locations, *regardless of how big or small the number may be!* If you view a stored real number in hex, binary, or even BCD, the number shown will be very difficult to decipher. Just like all other number types, you must keep track of real number locations in memory, so they can be read with the proper instructions later.

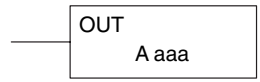
The previous example above stored a real number in V1400 and V1401. Suppose that now we want to retrieve that number. Just use the Load Real with the V data type, as shown to the right. Next we could perform real math on it, or convert it to a binary number.



Out (OUT)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Out instruction is a 16-bit instruction that copies the value in the lower 16 bits of the accumulator to a specified V-memory location (Aaaa).



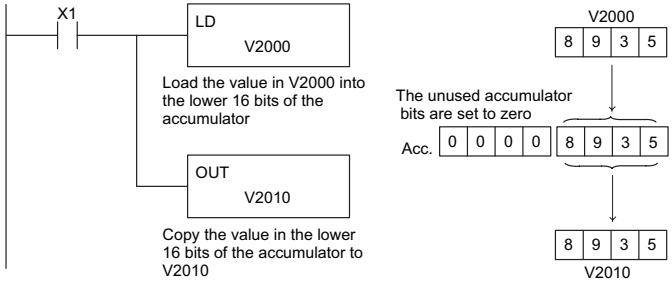
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All; see Memory map			
Pointer	P	–	All V-memory; see Memory map		

Discrete Bit Flags	Description
SP76	ON when the value loaded into the accumulator by any instruction is zero.

In the following example, when X1 is on, the value in V2000 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the lower 16 bits of the accumulator are copied to V2010 using the Out instruction.

DS	Used
HPP	Used

DirectSOFT



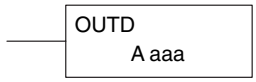
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→					
C 2	A 0	A 0	A 0	ENT				
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

Out Double (OUTD)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Out Double instruction is a 32-bit instruction that copies the value in the accumulator to two consecutive V-memory locations at a specified starting location (Aaaa).

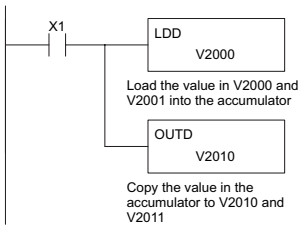


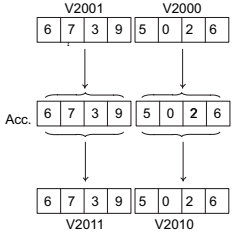
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All; see Memory map			
Pointer	P	–	All V-memory; see Memory map		

In the following example, when X1 is on, the 32-bit value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.

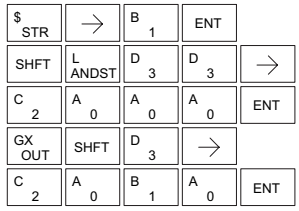
DS	Used
HPP	Used

***Direct*SOFT**





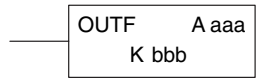
Handheld Programmer Keystrokes



Out Formatted (OUTF)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

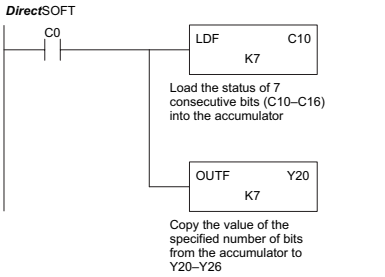
The Out Formatted instruction outputs 1 to 32 bits from the accumulator to the specified discrete memory locations. The instruction requires a starting location (Aaaa) for the destination and the number of bits (Kbbb) to be output.

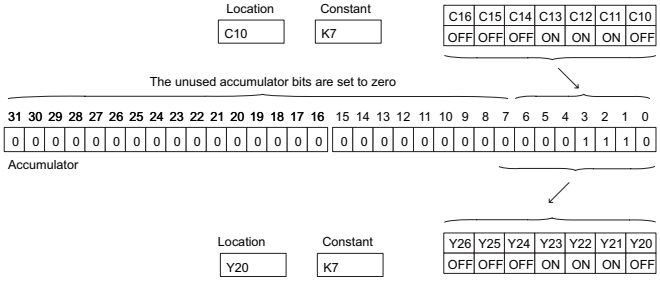


DS	Used
HPP	Used

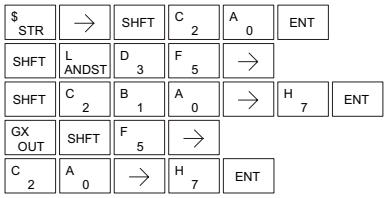
Operand Data Type		Range					
		D2-240		D2-250-1		D2-260/D2-262	
	A	aaa	bbb	aaa	bbb	aaa	bbb
Constant	K		1-32		1-32		1-32

In the following example, when C0 is on, the binary pattern of C10–C16 (7 bits) will be loaded into the accumulator using the Load Formatted instruction. The lower 7 bits of the accumulator are output to Y20–Y26 using the Out Formatted instruction.





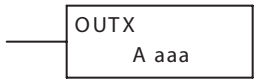
Handheld Programmer Keystrokes



Out Indexed (OUTX)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

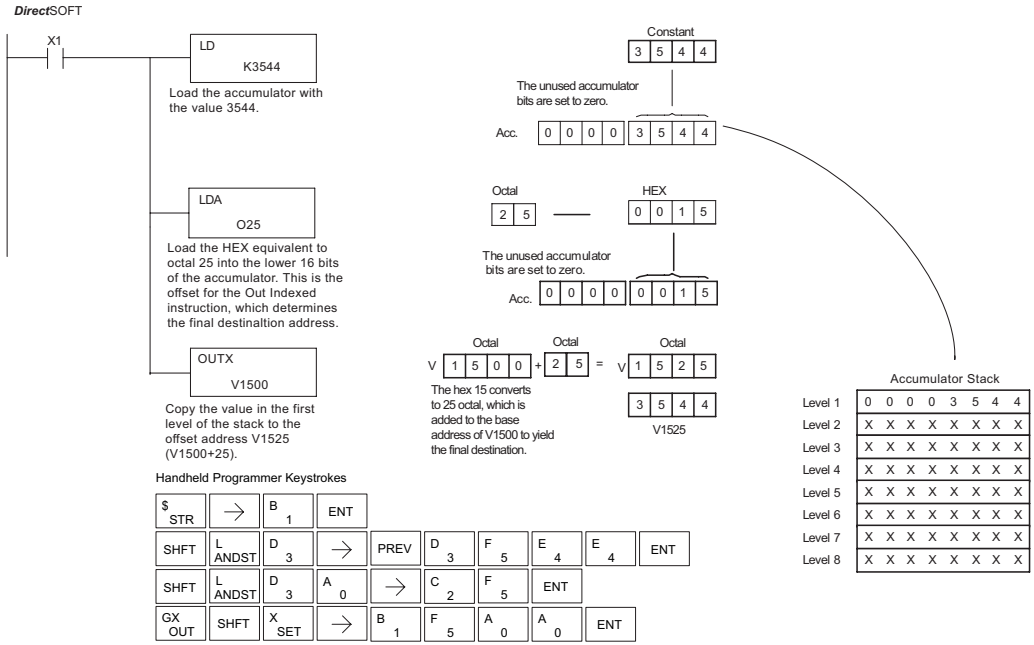
The Out Indexed instruction is a 16-bit instruction. It copies a 16-bit or 4-digit value from the first level of the accumulator stack to a source address offset by the value in the accumulator (V-memory + offset). This instruction interprets the offset value as a HEX number. The upper 16 bits of the accumulator are set to zero.



DS	Used
HPP	Used

Operand Data Type		Range	
		D2-250-1 Range	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All. See memory map	All. See memory map
Pointer	P	All V-memory. See memory map	All V-memory. See memory map

In the following example, when X1 is on, the constant value 3544 is loaded into the accumulator. This is the value that will be output to the specified offset V-memory location (V1525). The value 3544 will be placed onto the stack when the Load Address instruction is executed. Remember, two consecutive Load instructions places the value of the first load instruction onto the stack. The Load Address instruction converts octal 25 to HEX 15 and places the value in the accumulator. The Out Indexed instruction outputs the value 3544, which resides in the first level of the accumulator stack to V1525.

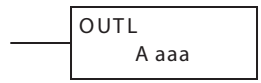


Out Least (OUTL)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

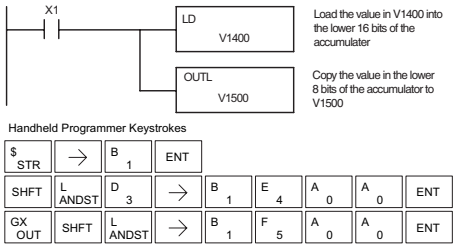
DS	Used
HPP	Used

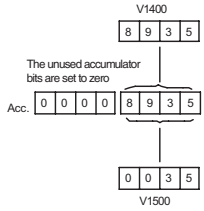
The Out Least instruction copies the value in the lower eight bits of the accumulator to the lower eight bits of the specified V-memory location (i.e., it copies the low byte of the low word of the accumulator).



In the following example, when X1 is on, the value in V1400 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the lower 8 bits of the accumulator are copied to V1500 using the Out Least instruction.

Operand Data Type		D2-260/D2-262 Range
	A	aaa
V-memory	V	All V-memory. See memory map
Pointer	P	All V-memory. See memory map





Out Most (OUTM)

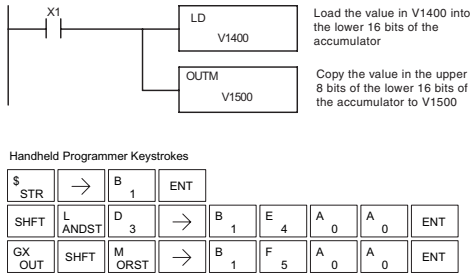
☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

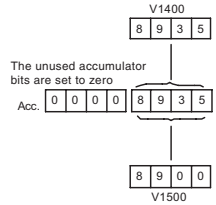
DS	Used
HPP	Used

The Out Most instruction copies the value in the upper eight l of the lower 16 bits of the accumulator to the upper eight l the specified V-memory location (i.e., it copies the high b the low word of the accumulator).

Operand Data Type		D2-260/D2-262 Range
	A	aaa
V-memory	V	All V-memory. See memory map
Pointer	P	All V-memory. See memory map

In the following example, when X1 is on, the value in V1400 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the upper 8 bits of the lower 16 bits of the accumulator are copied to V1500 using the Out Most instruction.





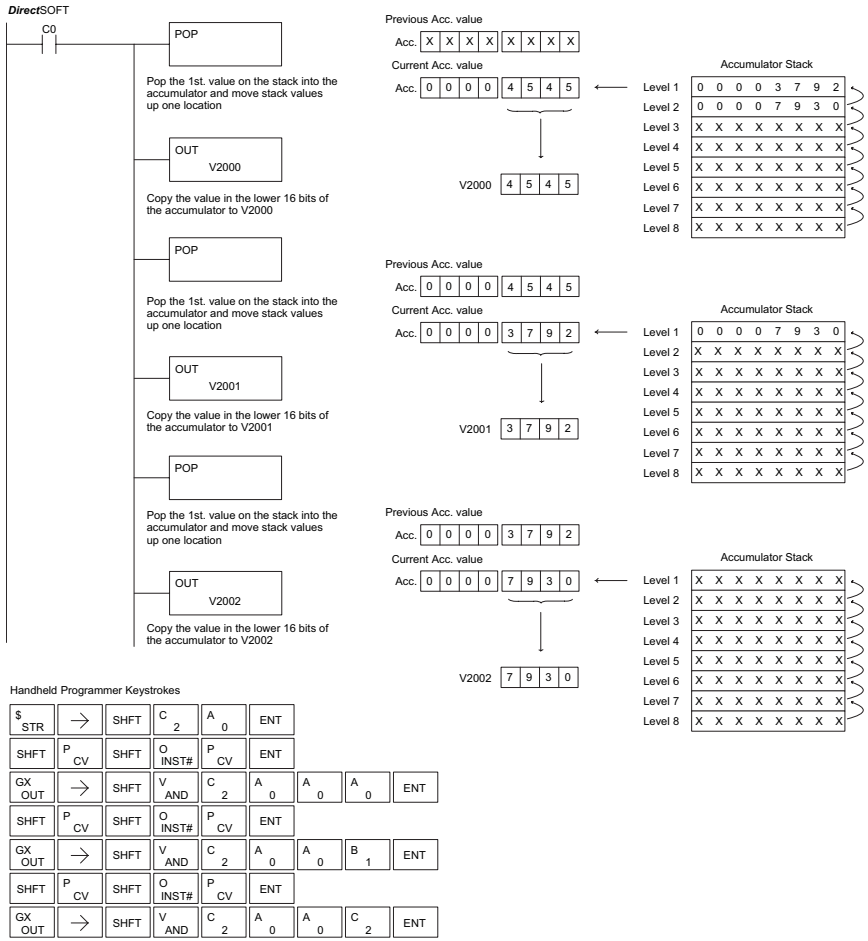
Pop (POP)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Pop instruction moves the value from the first level of the accumulator stack (32 bits) to the accumulator and shifts each value in the stack up one level. In the example below, when C0 is on, the value 4545 that was on top of the stack is moved into the accumulator using the Pop instruction. The value is output to V2000 using the Out instruction. The next Pop moves the value 3792 into the accumulator and outputs the value to V2001. The last Pop moves the value 7930 into the accumulator and outputs the value to V2002. Please note if the value in the stack were greater than 16 bits (4 digits) the Out Double instruction would be used and two V-memory locations for each Out Double must be allocated.

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.



Handheld Programmer Keystrokes

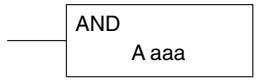
$ STR	→	SHFT	C 2	A 0	ENT			
SHFT	P CV	SHFT	O INST#	P CV	ENT			
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	A 0	ENT
SHFT	P CV	SHFT	O INST#	P CV	ENT			
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	B 1	ENT
SHFT	P CV	SHFT	O INST#	P CV	ENT			
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	C 2	ENT

Logical Instructions (Accumulator)

And (AND)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The And instruction is a 16-bit instruction that logically ANDs the value in the lower 16 bits of the accumulator with a specified V-memory location (Aaaa). The result resides in the accumulator. The discrete status flag indicates if the result of the And is zero.



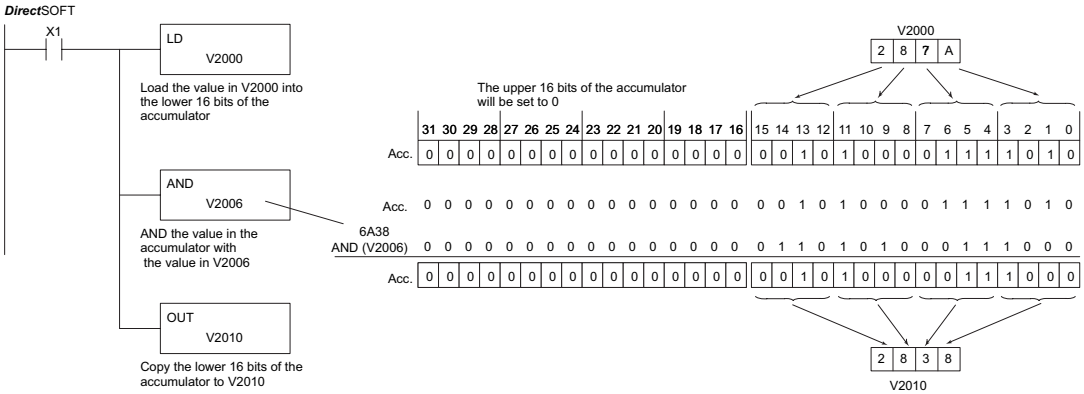
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All; see Memory map			
Pointer	P	–	All V-memory; see Memory map		

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.

***NOTE**: The status flags are only valid until another instruction that uses the same flags is executed.*

DS	Used
HPP	Used

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the accumulator is anded with the value in V2006 using the And instruction. The value in the lower 16 bits of the accumulator is output to V2010 using the Out instruction.



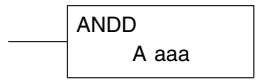
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT
V AND	→	SHFT	V AND	C 2	A 0	A 0	G 6	ENT
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

And Double (ANDD)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The And Double is a 32-bit instruction that logically ANDs the value in the accumulator with two consecutive V-memory locations or an 8-digit (max) constant value (Aaaa). The result resides in the accumulator. Discrete status flags indicate if the result of the And Double is zero or a negative number (the most significant bit is on).



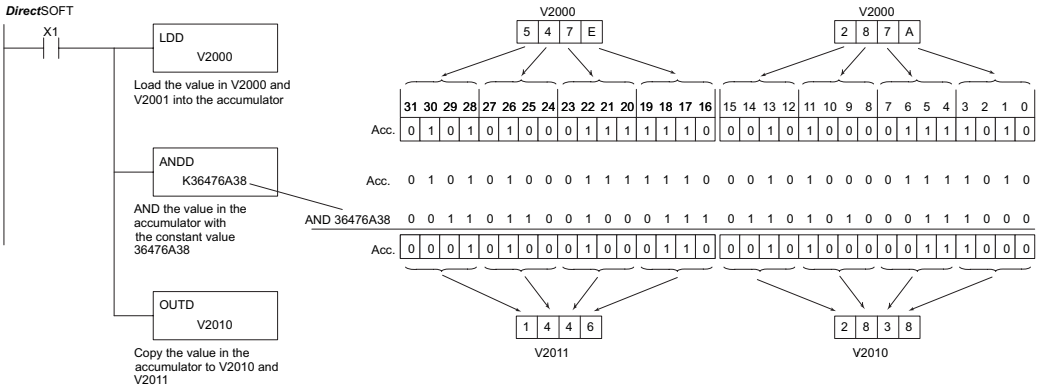
DS	Used
HPP	Used

Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	-	-	All. See memory map	
Pointer	P	-	-	All V-memory. See memory map	
Constant	K	0-FFFFFFFF	0-FFFFFFFF	0-FFFFFFFF	0-FFFFFFFF

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on if the result in the accumulator is negative

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is anded with 36476A38 using the And Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT																
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT										
V AND	SHFT	D 3	→	SHFT	K JMP	D 3	G 6	E 4	H 7	G 6	SHFT	A 0	SHFT	D 3	I 8	ENT			
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT											

And Formatted (ANDF)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The And Formatted instruction logically ANDs the binary value in the accumulator and a specified range of discrete memory bits (1 to 32). The instruction requires a starting location (Aaaa) and number of bits (Kbbb) to be ANDed. Discrete status flags indicate if the result is zero or a negative number (the most significant bit =1).

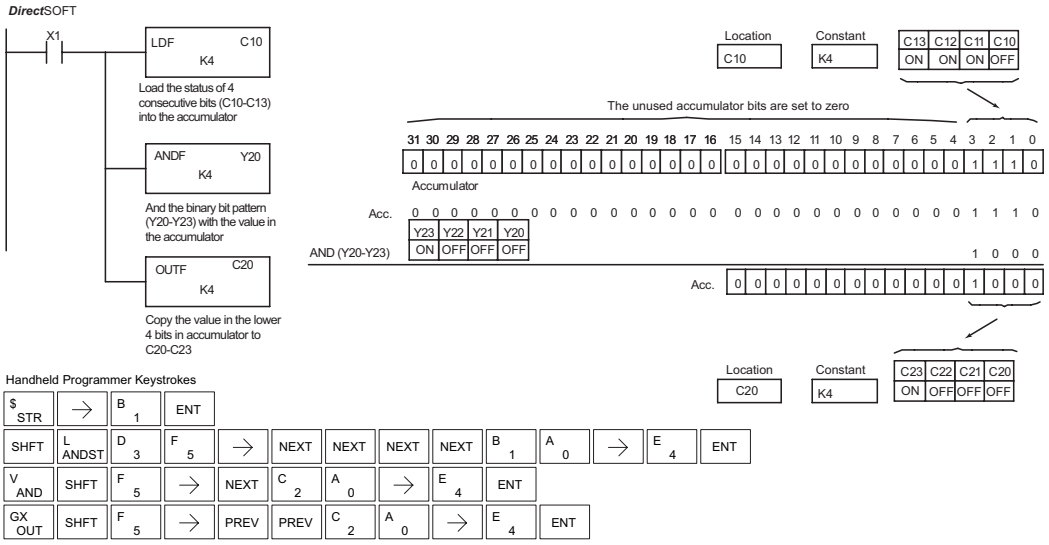
```
ANDF    A aaa
        K bbb
```

Operand Data Type		D2-250-1 Range		D2-260/D2-262 Range	
	A	aaa	bbb	aaa	bbb
Inputs	X	0–777	–	0–1777	–
Outputs	Y	0–777	–	0–1777	–
Control Relays	C	0–1777	–	0–3777	–
Stage bits	S	0–1777	–	0–1777	–
Timer bits	T	0–377	–	0–377	–
Counter bits	CT	0–177	–	0–377	–
Special Relays	SP	0-777	–	0–777	–
Global I/O	GX/GY	-	–	0-3777	–
Constant	K	-	1–32	-	1–32

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on if the result in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the Load Formatted instruction loads C10–C13 (4 binary bits) into the accumulator. The accumulator content is logically ANDed with the bit pattern from Y20–Y23 using the And Formatted instruction. The Out Formatted instruction outputs the accumulator's lower four bits to C20–C23.



And with Stack (ANDS)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

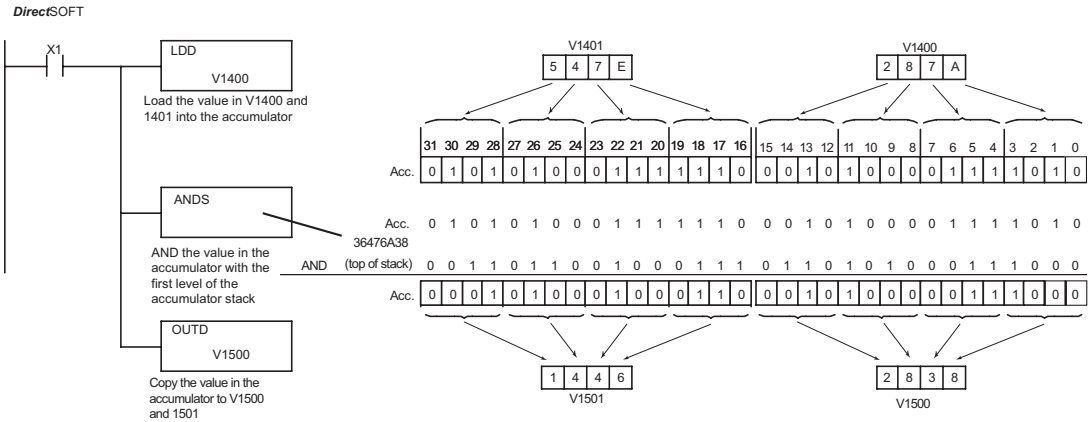
The And with Stack instruction is a 32-bit instruction that logically ANDs the value in the accumulator with the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed from the stack and all values are moved up one level. Discrete status flags indicate if the result of the And with Stack is zero or a negative number (the most significant bit is on).

DS	Used
HPP	Used

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on if the result in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the binary value in the accumulator will be anded with the binary value in the first level of the accumulator stack. The result resides in the accumulator. The 32-bit value is then output to V1500 and V1501.



Handheld Programmer Keystrokes

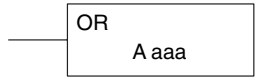
$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
V AND	SHFT	S RST	ENT						
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

Or (OR)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Or instruction is a 16-bit instruction that logically ORs the value in the lower 16 bits of the accumulator with a specified V-memory location (Aaaa). The result resides in the accumulator. The discrete status flag indicates if the result of the OR is zero.

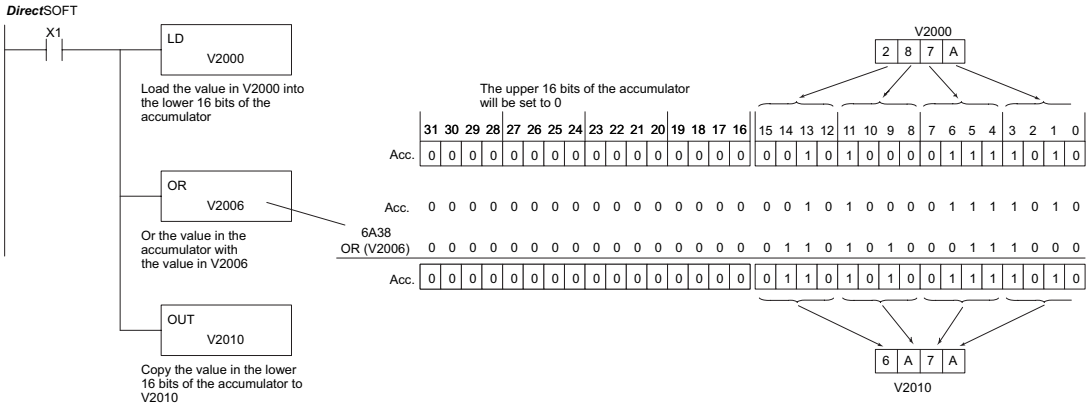


Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All; see Memory map			
Pointer	P	–	All V-memory; see Memory map		

Discrete Bit Flags	Description
SP63	Will be ON if the result in the accumulator is zero.

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the accumulator is OR'd with V2006 using the OR instruction. The value in the lower 16 bits of the accumulator are output to V2010 using the Out instruction.



Handheld Programmer Keystrokes

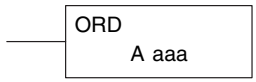
$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT
Q OR	→	SHFT	V AND	C 2	A 0	A 0	G 6	ENT
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

Or Double (ORD)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Or Double is a 32-bit instruction that ORs the value in the accumulator with the value (Aaaa) or an 8-digit (max) constant value. The result resides in the accumulator. Discrete status flags indicate if the result of the Or Double is zero or a negative number (the most significant bit is on).

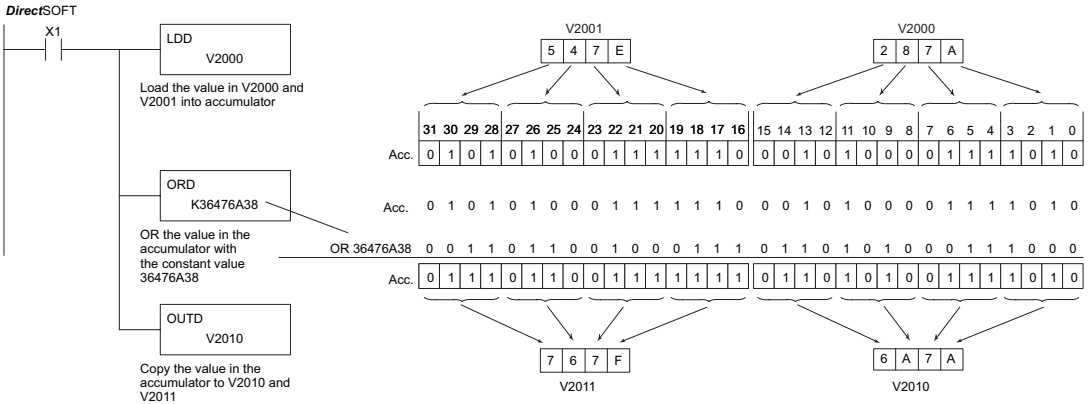


Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	–	–	All; see Memory map	
Pointer	P	–	–	All V-memory; see Memory map	
Constant	K	0-FFFFFFFF			

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on if the result in the accumulator is negative

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is OR'd with 36476A38 using the Or Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.



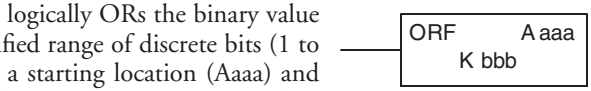
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT																					
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT															
Q OR	SHFT	D 3	→	SHFT	K JMP	D 3	G 6	E 4	H 7	G 6	SHFT	A 0	SHFT	D 3	I 8	ENT								
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT																

Or Formatted (ORF)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The Or Formatted instruction logically ORs the binary value in the accumulator and a specified range of discrete bits (1 to 32). The instruction requires a starting location (Aaaa) and the number of bits (Kbbb) to be ORed. Discrete status flags indicate if the result is zero or negative (the most significant bit =1).



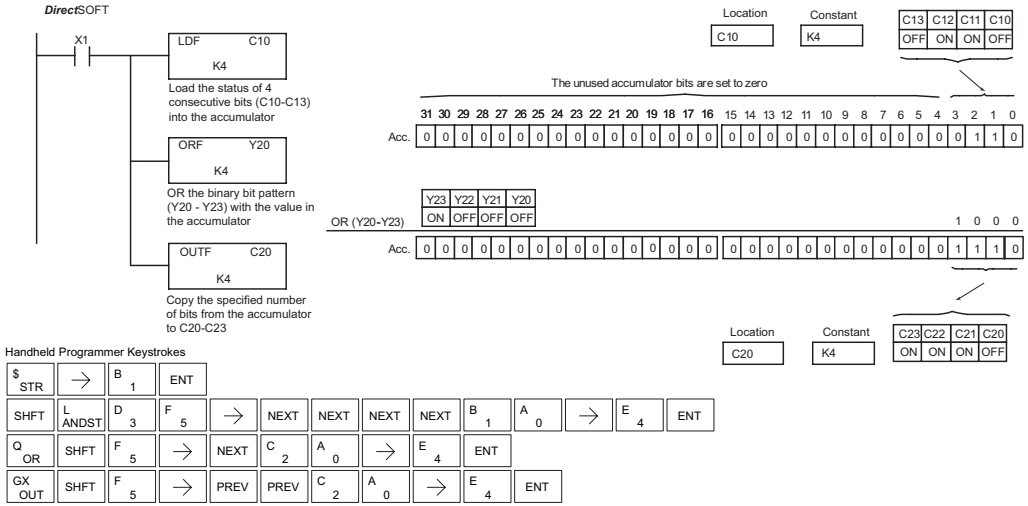
DS	Used
HPP	Used

Operand Data Type		D2-250-1 Range		D2-260/D2-262 Range	
	A	aaa	bbb	aaa	bbb
Inputs	X	0–777	–	0–1777	–
Outputs	Y	0–777	–	0–1777	–
Control Relays	C	0–1777	–	0–3777	–
Stage bits	S	0–1777	–	0–1777	–
Timer bits	T	0–377	–	0–377	–
Counter bits	CT	0–177	–	0–377	–
Special Relays	SP	0-777	–	0–777	–
Global I/O	GX/GY	-	–	0-3777	–
Constant	K	-	1–32	-	1–32

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on if the result in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on the Load Formatted instruction loads C10–C13 (4 binary bits) into the accumulator. The Or Formatted instruction logically ORs the accumulator contents with Y20–Y23 bit pattern. The Out Formatted instruction outputs the accumulator's lower four bits to C20–C23.



Or with Stack (ORS)

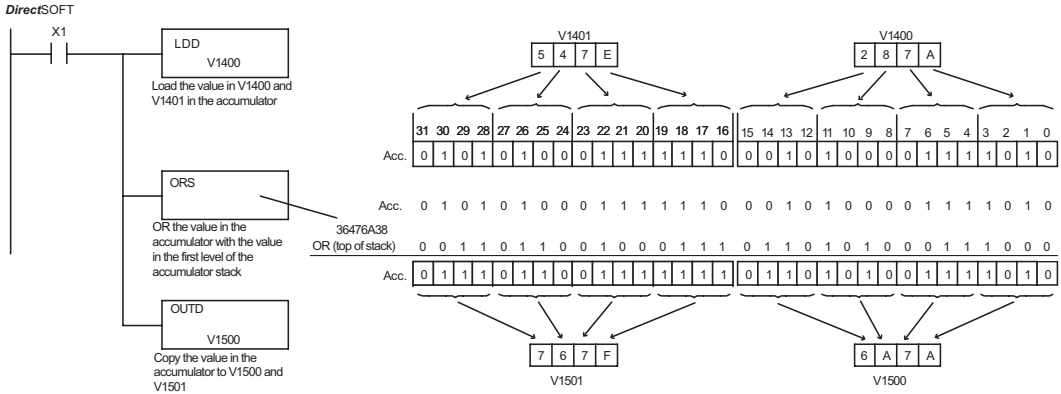
☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Or with Stack instruction is a 32-bit instruction that logically ORs the value in the accumulator with the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed from the stack and all values are moved up one level. Discrete status flags indicate if the result of the Or with Stack is zero or a negative number (the most significant bit is on).

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on if the result in the accumulator is negative

In the following example, when X1 is on, the binary value in the accumulator will be ORed with the binary value in the first level of the stack. The result resides in the accumulator.



Handheld Programmer Keystrokes

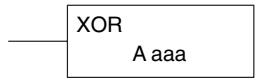
$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
Q OR	SHFT	S RST	ENT					
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT

Exclusive Or (XOR)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Exclusive Or instruction is a 16-bit instruction that performs an exclusive OR of the value in the lower 16 bits of the accumulator and a specified V-memory location (Aaaa). The result resides in the accumulator. The discrete status flag indicates if the result of the XOR is zero.

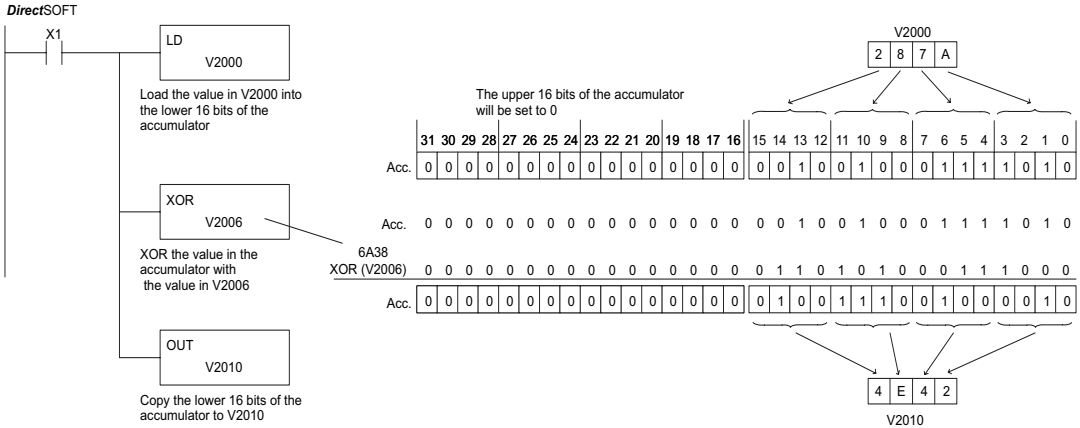


Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All; see Memory map			
Pointer	P	–	All V-memory; see Memory map		

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on if the result in the accumulator is negative

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the accumulator is exclusive OR'd with V2006 using the Exclusive Or instruction. The value in the lower 16 bits of the accumulator are output to V2010 using the Out instruction.



Handheld Programmer Keystrokes

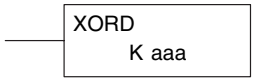
$ STR	→	B 1	ENT							
SHFT	L ANDST	D 3	→	SHFT	V AND	C 2	A 0	A 0	A 0	ENT
SHFT	X SET	SHFT	Q OR	→	SHFT	V AND	C 2	A 0	A 0	G 6	ENT
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT		

Exclusive Or Double (XORD)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Exclusive Or Double is a 32-bit instruction that performs an exclusive OR of the value in the accumulator and the value (Kaaa), which is an 8-digit (max) constant. The result resides in the accumulator. Discrete status flags indicate if the result of the Exclusive Or Double is zero or a negative number (the most significant bit is on).

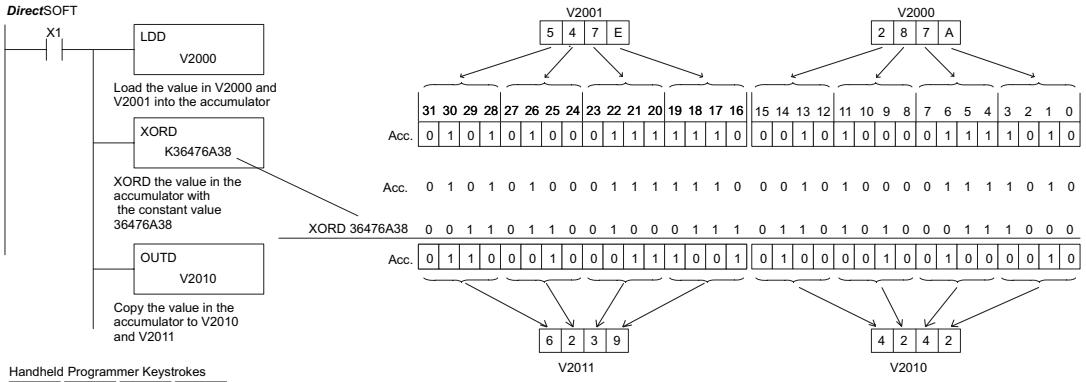


Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
Constant	K	0-FFFFFFFF	0-FFFFFFFF	0-FFFFFFFF	0-FFFFFFFF

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on if the result in the accumulator is negative

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is exclusively OR'd with 36476A38 using the Exclusive Or Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	X SET	Q OR	SHFT	D 3	→	SHFT	K JMP		
D 3	G 6	E 4	H 7	G 6	SHFT	A 0	SHFT	D 3	I 8	ENT
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT		

Exclusive OR Formatted (XORF)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Exclusive Or Formatted instruction performs an exclusive OR of the binary value in the accumulator and a specified range of discrete memory bits (1 to 32).

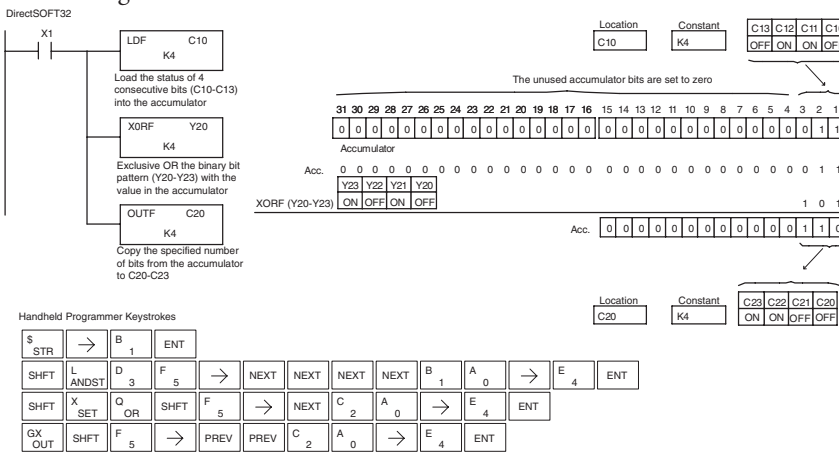
The instruction requires a starting location (Aaaa) and the number of bits (Kbbb) to be exclusive OR'd. Discrete status flags indicate if the result of the Exclusive Or Formatted is zero or negative (the most significant bit is on).

Operand Data Type		D2-250-1 Range		D2-260/D2-262 Range	
	A	aaa	bbb	aaa	bbb
Inputs	X	0–777	–	0–1777	–
Outputs	Y	0–777	–	0–1777	–
Control Relays	C	0–1777	–	0–3777	–
Stage bits	S	0–1777	–	0–1777	–
Timer bits	T	0–377	–	0–377	–
Counter bits	CT	0–177	–	0–377	–
Special Relays	SP	0-777	–	0–777	–
Global I/O	GX/GY	-	–	0-3777	–
Constant	K	-	1–32	-	1–32

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on if the result in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the binary pattern of C10–C13 (4 bits) will be loaded into the accumulator using the Load Formatted instruction. The value in the accumulator will be logically Exclusive OR'd with the bit pattern from Y20–Y23 using the Exclusive Or Formatted instruction. The value in the lower 4 bits of the accumulator are output to C20–C23 using the Out Formatted instruction.



Exclusive Or with Stack (XORS)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

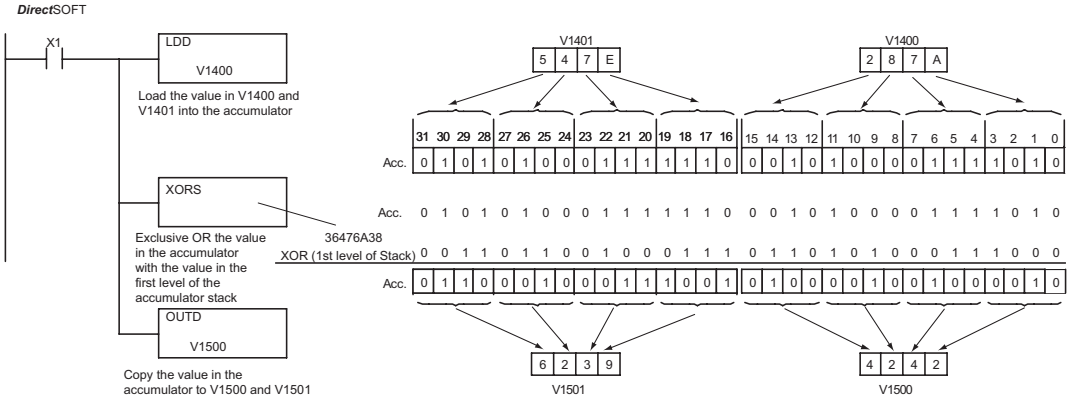
DS	Used
HPP	Used

The Exclusive Or with Stack instruction is a 32-bit instruction that performs an Exclusive Or of the value in the accumulator with the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed from the stack and all values are moved up one level. Discrete status flags indicate if the result of the Exclusive Or with Stack is zero or a negative number (the most significant bit is on).

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on if the result in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the binary value in the accumulator will be Exclusive OR'd with the binary value in the first level of the accumulator stack. The result will reside in the accumulator.



Handheld Programmer Keystrokes

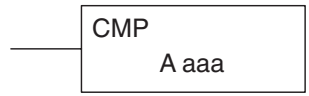
$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	X SET	Q OR	SHFT	S RST	ENT				
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

Compare (CMP)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The compare instruction is a 16-bit instruction that compares the value in the lower 16 bits of the accumulator with the value in a specified V-memory location (Aaaa). The corresponding status flag will be turned on indicating the result of the comparison.

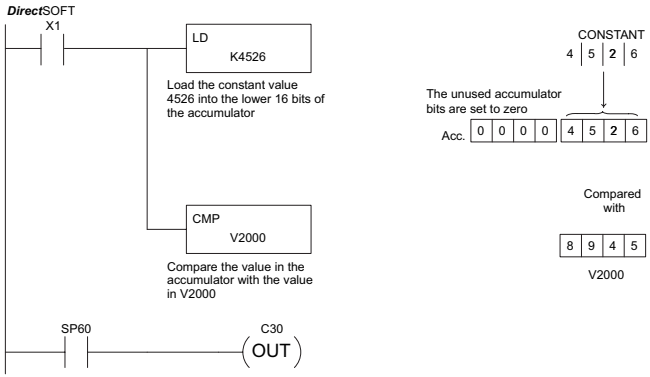


Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All; see Memory map			
Pointer	P	–	All V-memory; see Memory map		

Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when X1 is on, the constant 4526 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the accumulator is compared with the value in V2000 using the Compare instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on, energizing contact C30.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT							
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	F 5	C 2	G 6	ENT
SHFT	C 2	SHFT	M ORST	P CV	→	C 2	A 0	A 0	A 0	ENT
$ STR	→	SHFT	SP STRN	G 6	A 0	ENT				
GX OUT	→	SHFT	C 2	D 3	A 0	ENT				

Compare Double (CMPD)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Compare Double instruction is a 32–bit instruction that compares the value in the accumulator with the value (Aaaa), which is either two consecutive V-memory locations or an 8–digit (max) constant. The corresponding status flag will be turned on indicating the result of the comparison.

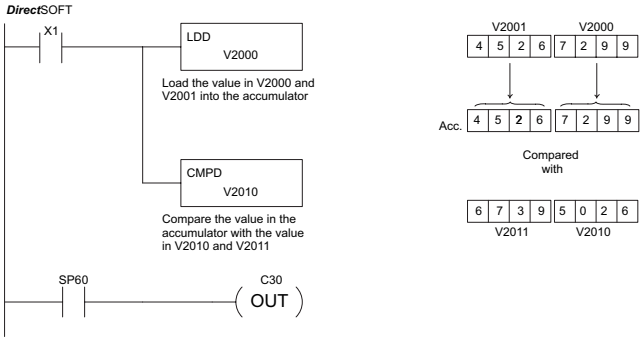
CMPD
A aaa

Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All; see Memory map			
Pointer	P	–	All V-memory; see Memory map		
Constant	K	0-FFFFFFFF	0-FFFFFFFF		

Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is compared with the value in V2010 and V2011 using the CMPD instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on, energizing contact C30.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT										
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT				
SHFT	C 2	SHFT	M ORST	P CV	D 3	→	C 2	A 0	B 1	A 0	ENT		
$ STR	→	SHFT	SP STRN	G 6	A 0	ENT							
GX OUT	→	SHFT	C 2	D 3	A 0	ENT							

Compare Formatted (CMPF)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The Compare Formatted compares the value in the accumulator with a specified number of discrete locations (1–32). The instruction requires a starting location (Aaaa) and the number of bits (Kbbb) to be compared. The corresponding status flag will be turned on indicating the result of the comparison.

```
CMPF    A aaa
        K bbb
```

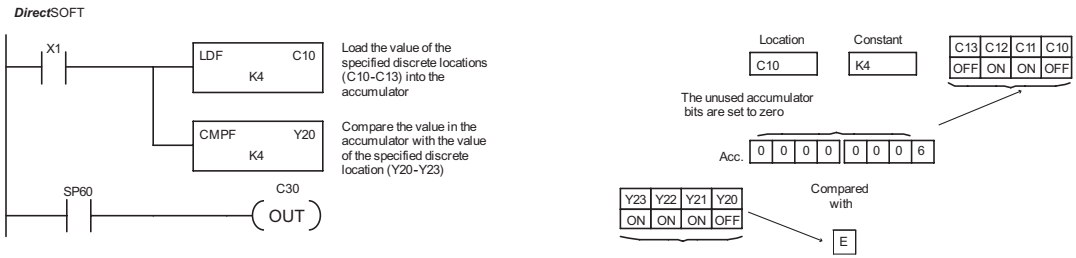
DS	Used
HPP	Used

Operand Data Type		Range			
		D2-250-1		D2-260/D2-262	
	A	aaa	bbb	aaa	bbb
Inputs	X	0–777	–	0–1777	–
Outputs	Y	0–777	–	0–1777	–
Control Relays	C	0–1777	–	0–3777	–
Stage bits	S	0–1777	–	0–1777	–
Timer bits	T	0–377	–	0–377	–
Counter bits	CT	0–177	–	0–377	–
Special Relays	SP	0-777	–	0–777	–
Global I/O	GX/GY	-	–	0-3777	–
Constant	K	-	1–32	-	1–32

Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the first level value in the Accumulator Stack.
SP61	On when the value in the accumulator is equal to the first level value in the Accumulator Stack
SP62	On when the value in the accumulator is greater than the first level value in the Accumulator Stack.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on the Load Formatted instruction loads the binary value (6) from C10–C13 into the accumulator. The CMPF instruction compares the value in the accumulator to the value in Y20–Y23 (E hex). The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on, energizing C30.



Compare with Stack (CMPS)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

The Compare with Stack instruction is a 32-bit instruction that compares the value in the accumulator with the value in the first level of the accumulator stack.

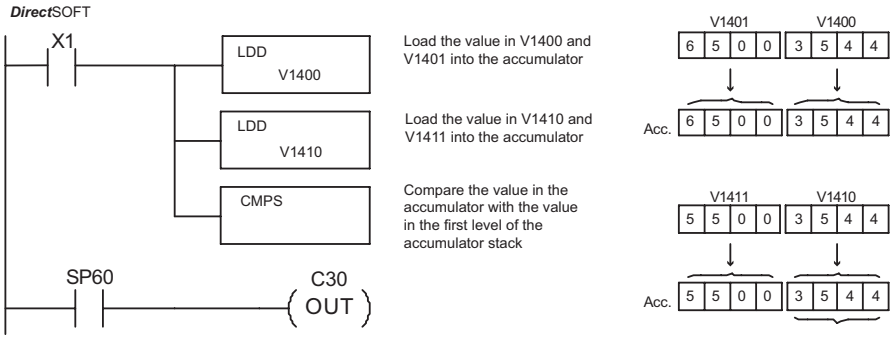
The corresponding status flag will be turned on indicating the result of the comparison. This does not affect the value in the accumulator.

DS	Used
HPP	Used

Discrete Bit Flags	Description
SP60	On when the value in the Accumulator is less than the first level value in the Accumulator Stack
SP61	On when the value in the Accumulator is equal to the first level value in the Accumulator Stack
SP62	On when the value in the Accumulator is greater than the first level value in the Accumulator Stack

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example when X1 is on, the value in V1400 and V1401 is loaded into the accumulator using the Load Double instruction. The value in V1410 and V1411 is loaded into the accumulator using the Load Double instruction. The value that was loaded into the accumulator from V1400 and V1401 is placed on top of the stack when the second Load instruction is executed. The value in the accumulator is compared with the value in the first level of the accumulator stack using the CMPS instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value in the stack, SP60 will turn on, energizing C30.



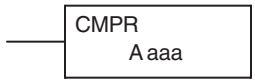
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	B 1	A 0	ENT
SHFT	C 2	SHFT	M ORST	P CV	S RST	ENT			
$ STR	→	SHFT	SP STRN	G 6	A 0	ENT			
GX OUT	→	SHFT	C 2	D 3	A 0	ENT			

Compare Real Number (CMPR)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The Compare Real Number instruction compares a real number value in the accumulator with two consecutive V-memory locations containing a real number. The corresponding status flag will be turned on indicating the result of the comparison. Both numbers being compared are 32 bits long.



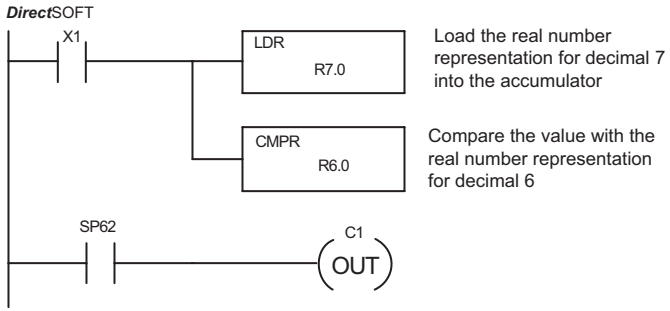
DS	Used
HPP	N/A

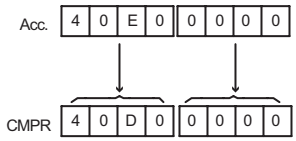
Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All. See memory map	All. See memory map
Pointer	P	All V-memory. See memory map	All V-memory. See memory map
Constant	R	-3.402823E+038 to + 3.402823E+038	-3.402823E+038 to + 3.402823E+038

Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.
SP71	On anytime the V-memory specified by a pointer (P) is not valid
SP75	On when a real number instruction is executed and a non-real number encountered.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the LDR instruction loads the real number representation for 7 decimal into the accumulator. The CMPR instruction compares the accumulator contents with the real representation for decimal 6. Since 7 > 6, the corresponding discrete status flag is turned on (special relay SP62).





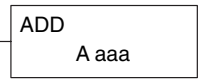
Math Instructions

Add (ADD)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

Add is a 16-bit instruction that adds a BCD value in the accumulator with a BCD value in a V-memory location (Aaaa). (You cannot use a constant (K) as the BCD value in the box.) The result resides in the accumulator.

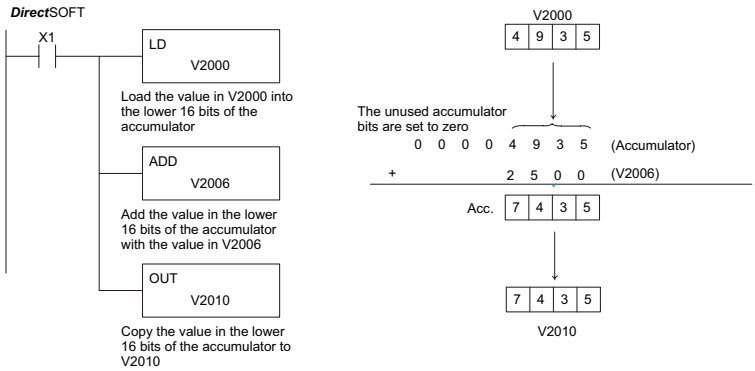


Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All; see Memory map			
Pointer	P	–	All V-memory; see Memory map		

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP66	On when the 16-bit addition instruction results in a carry
SP67	On when the 32-bit addition instruction results in a carry
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number is encountered

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the lower 16 bits of the accumulator are added to the value in V2006 using the Add instruction. The value in the accumulator is copied to V2010 using the Out instruction.



Handheld Programmer Keystrokes

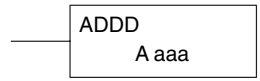
$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	A 0	D 3	D 3	→	C 2	A 0	A 0	G 6	ENT
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

Add Double (ADDD)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

Add Double is a 32-bit instruction that adds the BCD value in the accumulator with a BCD value (Aaaa), which is either two consecutive V-memory locations or an 8–digit (max) BCD constant. The result resides in the accumulator.

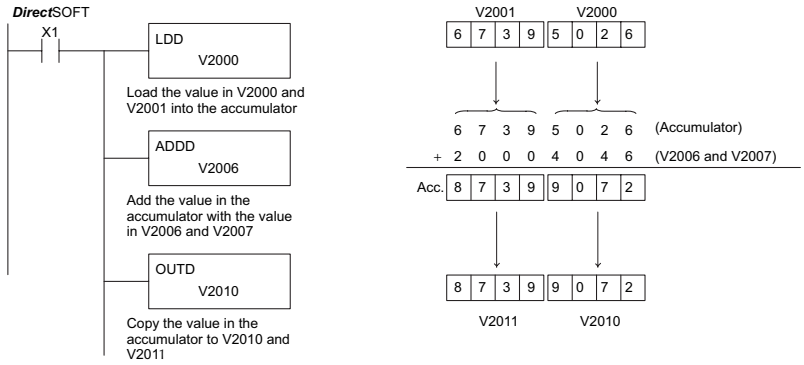


Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All; see Memory map			
Pointer	P	–	All V-memory; see Memory map		
Constant	K	0-99999999	0-99999999		

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP66	On when the 16-bit addition instruction results in a carry
SP67	On when the 32-bit addition instruction results in a carry
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number is encountered

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is added with the value in V2006 and V2007 using the Add Double instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

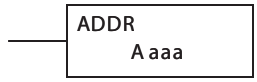
$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	A 0	D 3	D 3	D 3	→	C 2	A 0	A 0	G 6	ENT
GX OUT	SHFT	D 3	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

Add Real (ADDR)

☐ 230
☐ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

Add Real is a 32-bit instruction that adds a real number, which is either two consecutive V-memory locations or a 32-bit constant, to a real number in the accumulator. Both numbers must conform to the IEEE floating point format. The result is a 32-bit real number that resides in the accumulator.

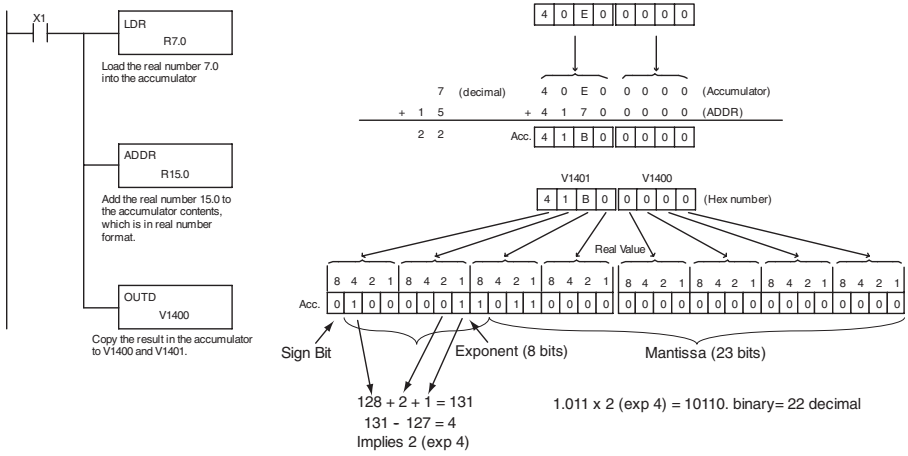


Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All. See memory map	All. See memory map
Pointer	P	All V-memory. See memory map	All V-memory. See memory map
Constant	R	-3.402823E+038 to + 3.402823E+038	-3.402823E+038 to + 3.402823E+038

Discrete Bit Flags Description	
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP71	On anytime the V-memory specified by a pointer (P) is not valid
SP72	On anytime the value in the accumulator is an invalid floating point number
SP73	On when a signed addition or subtraction results in a incorrect sign bit
SP74	On anytime a floating point math operation results in an underflow error
SP75	On when a real number instruction is executed and a non-real number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

DirectSOFT



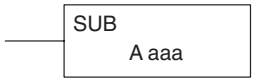
NOTE 1: *The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use* ***Direct****SOFT for this feature.*
NOTE 2: *If the value being added to a real number is 16,777,216 times smaller than the real number, the calculation will not work.*

Subtract (SUB)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

Subtract is a 16-bit instruction that subtracts the BCD value (Aaaa) in a V-memory location from the BCD value in the lower 16 bits of the accumulator. The result resides in the accumulator.

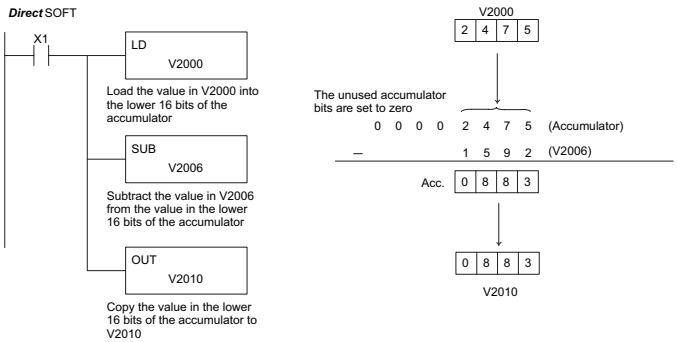


Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All; see Memory map			
Pointer	P	–	All V-memory; see Memory map		

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP66	On when the 16-bit addition instruction results in a carry
SP67	On when the 32-bit addition instruction results in a carry
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number is encountered

NOTE: *A constant (K) cannot be used for the BCD value.*
Status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in V2006 is subtracted from the value in the accumulator using the Subtract instruction. The value in the accumulator is copied to V2010 using the Out instruction.



Subtract Double (SUBD)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

Subtract Double is a 32-bit instruction that subtracts the BCD value (Aaaa), which is either two consecutive V-memory locations or an 8-digit (max) constant, from the BCD value in the accumulator. The result resides in the accumulator.

SUBD
A aaa

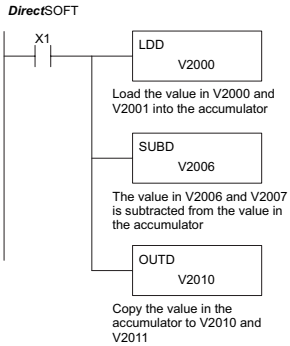
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All (See page 3 - 54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)
Pointer	P	-	All V-memory (See page 3-55)	All V-memory (See page 3-56)	All V-memory (See page 3-57)
Constant	K	0-99999999	0-99999999	0-99999999	0-99999999

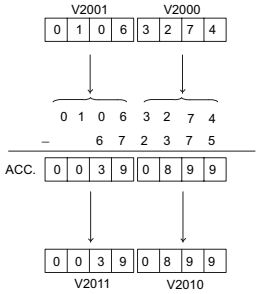
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP66	On when the 16-bit addition instruction results in a carry
SP67	On when the 32-bit addition instruction results in a carry
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number is encountered

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

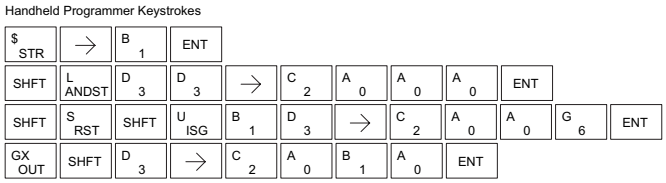
In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in V2006 and V2007 is subtracted from the value in the accumulator. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.

DS	Used
HPP	Used





Handheld Programmer Keystrokes



Subtract Real (SUBR)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

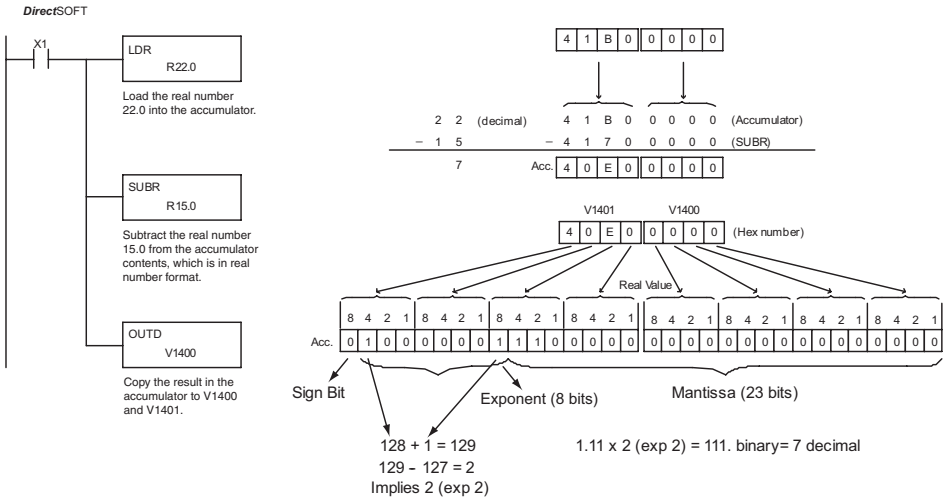
The Subtract Real is a 32-bit instruction that subtracts a real number, which is either two consecutive V-memory locations or a 32-bit constant, from a real number in the accumulator. The result is a 32-bit real number that resides in the accumulator. Both numbers must be Real data type (IEEE floating point format).

SUBR
A aaa

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All. (See page 3-56)	All. (See page 3-57)
Pointer	P	All V-memory (See page 3-56)	All V-memory (See page 3-57)
Constant	R	-3.402823E+038 to + 3.402823E+038	-3.402823E+038 to + 3.402823E+038

Discrete Bit Flags Description	
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP71	On anytime the V-memory specified by a pointer (P) is not valid
SP72	On anytime the value in the accumulator is an invalid floating point number
SP73	On when a signed addition or subtraction results in a incorrect sign bit
SP74	On anytime a floating point math operation results in an underflow error
SP75	On when a real number instruction is executed and a non-real number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

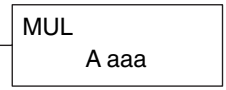


NOTE: *The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use* ***Direct****SOFT for this feature.*

Multiply (MUL)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

Multiply is a 16-bit instruction that multiplies the BCD value (Aaaa), which is either a V-memory location or a 4–digit (max) constant, by the BCD value in the lower 16 bits of the accumulator The result can be up to 8 digits and resides in the accumulator.



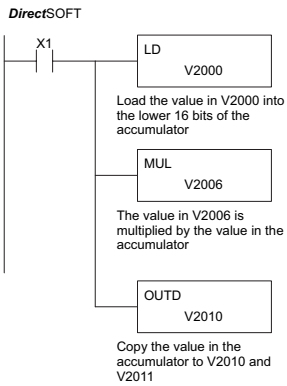
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All (See page 3 - 54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)
Pointer	P	-	All V-memory (See page 3-55)	All V-memory (See page 3-56)	All V-memory (See page 3-57)
Constant	K	0-9999	0-99999999	0-9999	0-9999

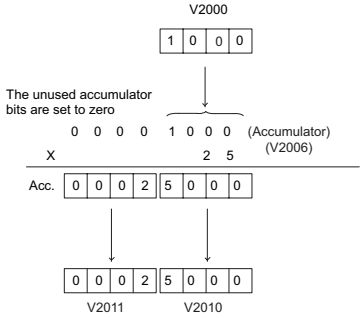
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number is encountered

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in V2006 is multiplied by the value in the accumulator. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.

DS	Used
HPP	Used





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT	
SHFT	M ORST	U ISG	L ANDST	→	C 2	A 0	A 0	G 6	ENT
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

Multiply Double (MULD)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

Multiply Double is a 32-bit instruction that multiplies the 8-digit BCD value in the accumulator by the 8-digit BCD value in the two consecutive V-memory locations specified in the instruction. The lower 8 digits of the results reside in the accumulator. Upper digits of the result reside in the accumulator stack.

MULD
A aaa

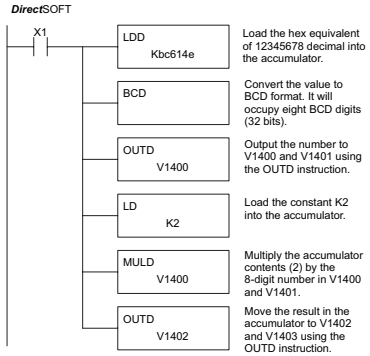
Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All V-mem (See page 3-56)	All V-mem (See page 3-57)
Pointer	P	All V-mem (See page 3-56)	All V-mem (See page 3-57)

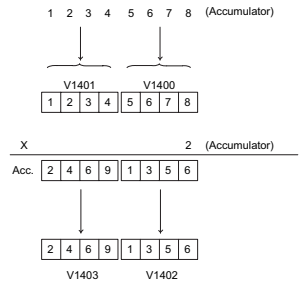
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number is encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the constant Kbc614e hex will be loaded into the accumulator. When converted to BCD the number is "12345678". That number is stored in V1400 and V1401. After loading the constant K2 into the accumulator, we multiply it times 12345678, which is 24691356.

DS	Used
HPP	Used





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT																
SHFT	L ANDST	D 3	D 3	→	PREV	SHFT	B 1	C 2	SHFT	G 6	B 1	E 4	SHFT	E 4	ENT				
SHFT	B 1	C 2	D 3	ENT															
GX OUT	SHFT	D 3	→	B 1	E 4	A 0	A 0	ENT											
SHFT	L ANDST	D 3	→	PREV	C 2	ENT													
SHFT	M ORST	U ISG	L ANDST	D 3	→	B 1	E 4	A 0	A 0	ENT									
GX OUT	SHFT	D 3	→	B 1	E 4	A 0	C 2	ENT											

Multiply Real (MULR)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

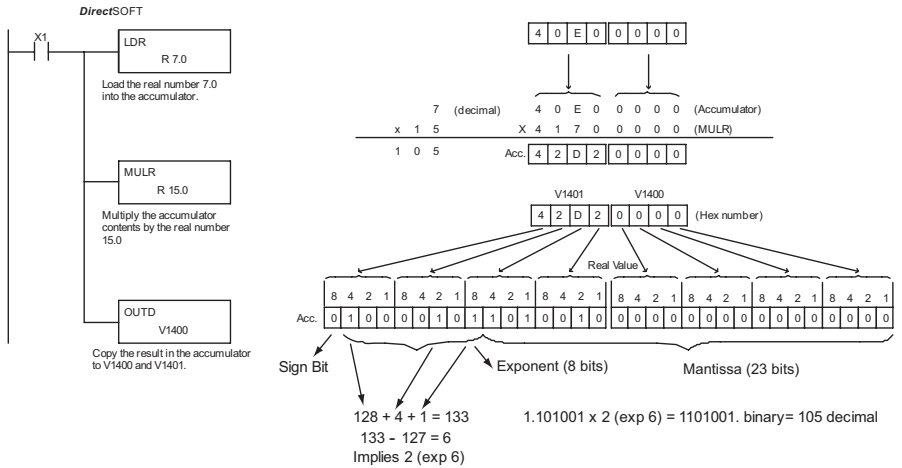
The Multiply Real instruction multiplies a real number in the accumulator with either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must be Real data type (IEEE floating point format).

MULR
A aaa

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All. (See page 3-56)	All. (See page 3-57)
Pointer	P	All V-memory (See page 3-56)	All V-memory (See page 3-57)
Constant	R	-3.402823E+038 to + 3.402823E+038	-3.402823E+038 to + 3.402823E+038

Discrete Bit Flags Description	
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP71	On anytime the V-memory specified by a pointer (P) is not valid
SP72	On anytime the value in the accumulator is an invalid floating point number
SP73	On when a signed addition or subtraction results in a incorrect sign bit
SP74	On anytime a floating point math operation results in an underflow error
SP75	On when a real number instruction is executed and a non-real number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

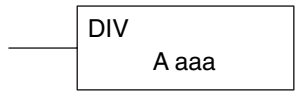


NOTE: *The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use* ***Direct****SOFT for this feature.*

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

Divide (DIV)

Divide is a 16-bit instruction that divides the BCD value in the accumulator by a BCD value (Aaaa), which is either a V-memory location or a 4-digit (max) constant. The first part of the quotient resides in the accumulator, and the remainder resides in the first stack location.



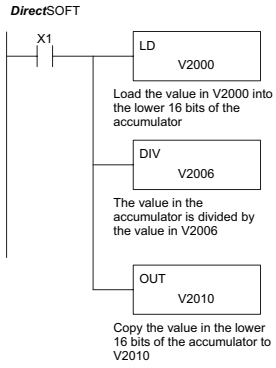
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All (See page 3-54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)
Pointer	P	-	All V-memory (See page 3-55)	All V-memory (See page 3-56)	All V-memory (See page 3-57)
Constant	K	1-9999	1-9999	1-9999	1-9999

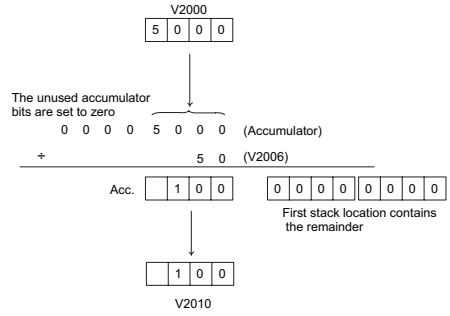
Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the accumulator will be divided by the value in V2006 using the Divide instruction. The value in the accumulator is copied to V2010 using the Out instruction.

DS	Used
HPP	Used





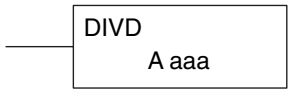
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	D 3	I 8	V AND	→	C 2	A 0	A 0	G 6	ENT
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

Divide Double (DIVD)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

Divide Double is a 32-bit instruction that divides the BCD value in the accumulator by a BCD value (Aaaa), which must be obtained from two consecutive V-memory locations (You cannot use a constant as the parameter in the box). The first part of the quotient resides in the accumulator, and the remainder resides in the first stack location.



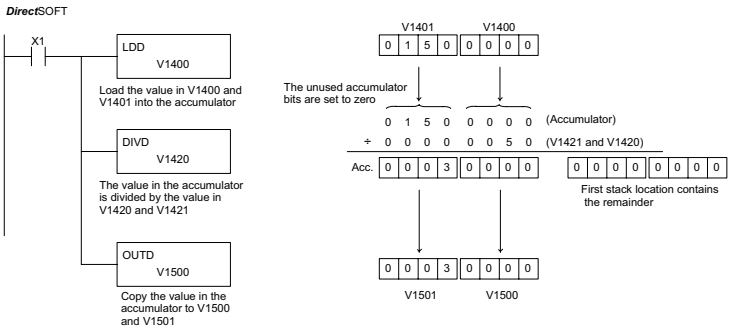
DS	Used
HPP	Used

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All V-memory (See page 3-56)	All V-memory (See page 3-57)
Pointer	P	All V-memory (See page 3-56)	All V-memory (See page 3-57)

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is divided by the value in V1420 and V1421 using the Divide Double instruction. The first part of the quotient resides in the accumulator and the remainder resides in the first stack location. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT							
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT	
SHFT	D 3	I 8	V AND	→	B 1	E 4	C 2	A 0	ENT	
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT		

Divide Real (DIVR)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

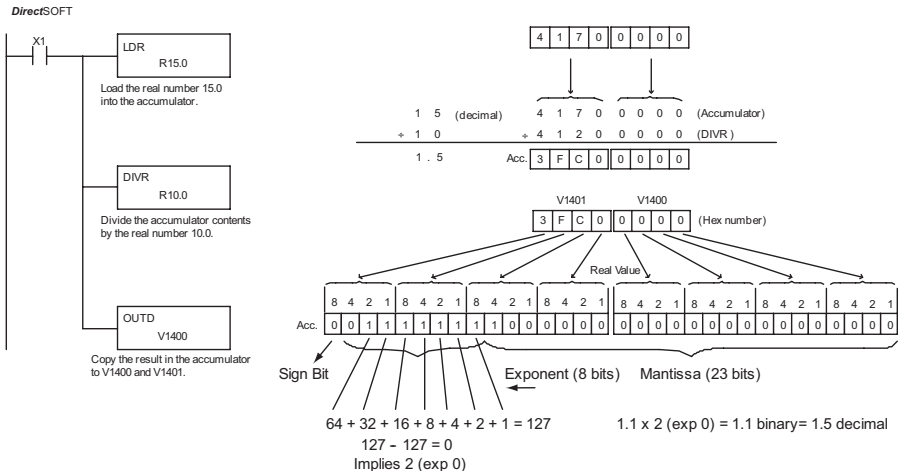
The Divide Real instruction divides a real number in the accumulator by either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must conform to the IEEE floating point format.

```
DIVR
   A aaa
```

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All (See page 3-56)	All (See page 3-57)
Pointer	P	All V-mem (See page 3-56)	All V-mem (See page 3-57)
Constant	R	-3.402823E + 038 to + 3.402823E + 038	-3.402823E + 038 to + 3.402823E + 038

Discrete Bit Flags Description	
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP71	On anytime the V-memory specified by a pointer (P) is not valid
SP72	On anytime the value in the accumulator is an invalid floating point number
SP73	On when a signed addition or subtraction results in a incorrect sign bit
SP74	On anytime a floating point math operation results in an underflow error
SP75	On when a real number instruction is executed and a non-real number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*



NOTE: *The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use* ***Direct****SOFT for this feature.*

Increment (INC)

The Increment instruction increments a BCD value in a specified V-memory location by "1" each time the instruction is executed.

INC
A aaa

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

Decrement (DEC)

The Decrement instruction decrements a BCD value in a specified V-memory location by "1" each time the instruction is executed.

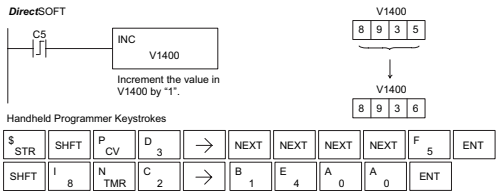
DEC
A aaa

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All V mem (See page 3-56)	All V-mem (See page 3-57)
Pointer	P	All V mem (See page 3-56)	All V-mem (See page 3-57)

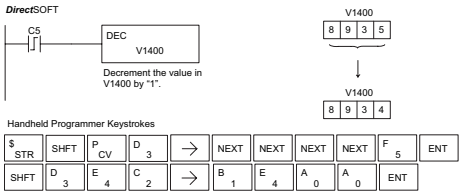
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following increment example, the value in V1400 increases by one each time that C5 is closed (true).



In the following decrement example, the value in V1400 is decreased by one each time that C5 is closed (true).



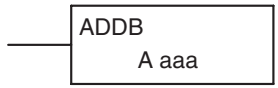
NOTE: *Use a pulsed contact closure to INC/DEC the value in V–memory once per closure.*

Add Binary (ADDB)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Add Binary instruction adds a 16-bit number (Aaaa) to the value stored in the accumulator. The number in the accumulator can be up to 32 bits long. The source of the 16-bit operand can be a constant or a data value located in V-memory. Add Binary performs the addition operation on the full binary representation of the operands, which distinguishes it from the Add instruction (see page 5-88), which treats the operands as BCD numbers. Although the addition operation is performed on the underlying binary values, the native display format is hexadecimal. For that reason you will need to load constants in hex.



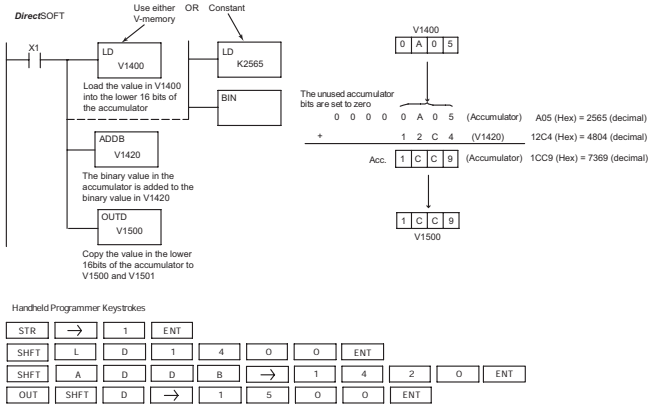
The sum of the Add Binary operation occupies the full 32-bit accumulator and requires an Out Double to move the sum to V-memory. If the value in the accumulator occupies fewer than 32 bits, leading zeros are loaded in the left-most empty bit positions.

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All (See page 3-56)	All (See page 3-57)
Pointer	P	All V-mem (See page 3-56)	All V-mem (See page 3-57)
Constant	K	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP66	On when the 16-bit addition instruction results in a carry
SP67	On when the 32-bit addition instruction results in a carr
SP70	On anytime the value in the accumulator is negative
SP73	On when a signed addition or subtraction results in an incorrect sign bit.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The binary value in V1420 is added to the binary value in the accumulator using the Add Binary instruction. The value in the accumulator is copied to V1500 - V1501 using the Out Double instruction.

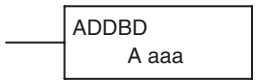


Add Binary Double (ADDBD)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

Add Binary Double is a 32-bit instruction that adds the binary value in the accumulator with the value (Aaaa), which is either two consecutive V-memory locations or an 8-digit (max.) binary constant. The result resides in the accumulator.

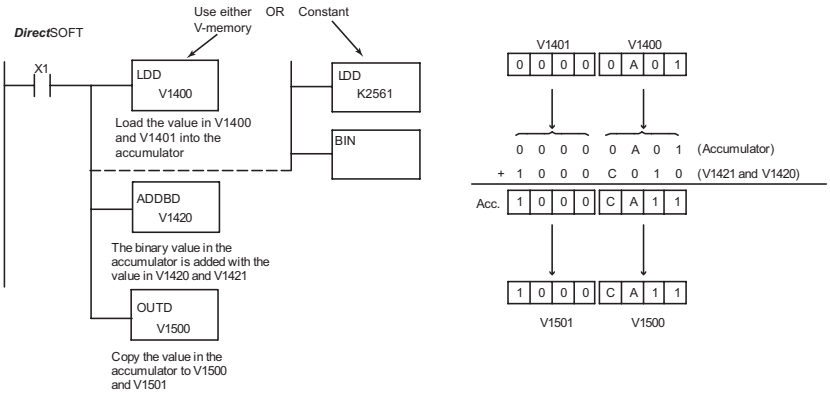


Operand Data Type		Range D2-260/D2-262
	A	**aaa**
V-memory	V	All (See page 3-57)
Pointer	P	All V mem (See page 3-57)
Constant	K	0-FFFFFFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP66	On when the 16-bit addition instruction results in a carry
SP67	On when the 32-bit addition instruction results in a carry
SP70	On anytime the value in the accumulator is negative
SP73	On when a signed addition or subtraction results in an incorrect sign bit

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The binary value in the accumulator is added with the binary value in V1420 and V1421 using the Add Binary Double instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

STR	1	↵							
LD	SHFT	D	SHFT	1	4	0	0	↵	
ADD	SHFT	B	D	SHFT	1	4	2	0	↵
OUT	SHFT	D	SHFT	1	5	0	0	↵	

Subtract Binary (SUBB)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

SUBB
A aaa

The Subtract Binary instruction subtracts a 16-bit number (Aaaa) from the value stored in the accumulator. The number in the accumulator can be up to 32 bits long. The source of the 16-bit operand can be a constant or a data value located in V-memory. Subtract Binary performs the subtraction operation on the full binary representation of the operands, which distinguishes it from the Subtract instruction (see page 5-91), which treats the operands as BCD numbers. Although the subtraction operation is performed on the underlying binary values, the native display format is hexadecimal. For that reason, you will need to load constants in hex.

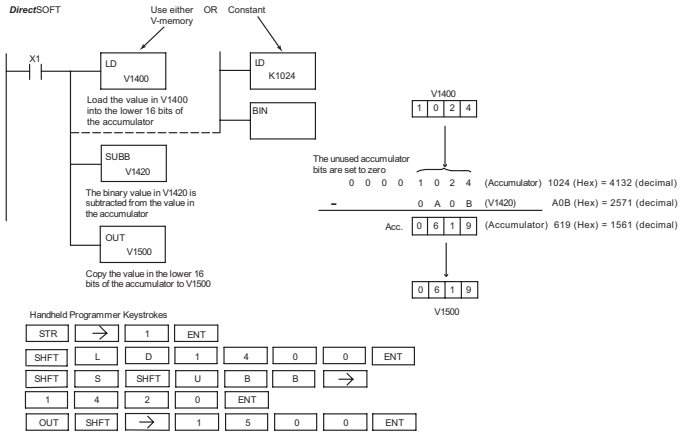
The difference (result) of the Subtract Binary operation occupies the full 32 bits of the accumulator and requires an Out Double to move the value to V-memory. If the value in the accumulator occupies fewer than 32 bits, leading zeros are loaded in the left-most empty bit positions of the accumulator.

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All (See page 3-56)	All (See page 3-57)
Pointer	P	All V-mem (See page 3-56)	All V-mem (See page 3-57)
Constant	K	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP64	On when the 16-bit subtraction instruction results in a borrow
SP65	On when the 32-bit subtraction instruction results in a borrow
SP70	On anytime the value in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The binary value in V1420 is subtracted from the binary value in the accumulator using the Subtract Binary instruction. The value in the accumulator is copied to V1500 - V1501 using the Out Double instruction.



Subtract Binary Double (SUBBD)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

Subtract Binary Double is a 32-bit instruction that subtracts the binary value (Aaaa), which is either two consecutive V-memory locations or an 8-digit (max) binary constant, from the binary value in the accumulator. The result resides in the accumulator.

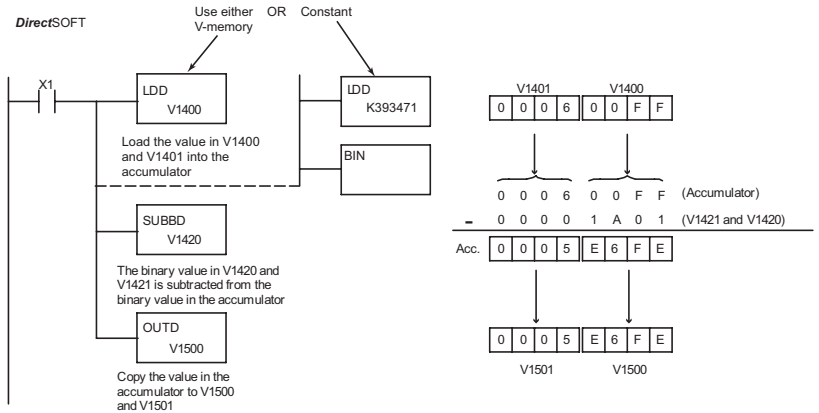
SUBBD
A aaa

Operand Data Type		Range D2-260/D2-262
	A	aaa
V-memory	V	All (See page 3-57)
Pointer	P	All V-mem (See page 3-57)
Constant	K	0-FFFFFFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP64	On when the 16-bit subtraction instruction results in a borrow
SP65	On when the 32-bit subtraction instruction results in a borrow
SP70	On anytime the value in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The binary value in V1420 and V1421 is subtracted from the binary value in the accumulator using the Subtract Binary Double instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



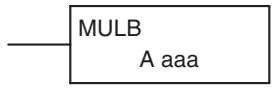
Handheld Programmer Keystrokes

STR	→	1	ENT						
SHFT	L	D	D	→	1	4	0	0	ENT
SHFT	S	SHFT	U	B	B	D	→		
1	4	2	0	ENT					
OUT	SHFT	D	→	1	5	0	0	ENT	

Multiply Binary (MULB)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used



The Multiply Binary instruction multiplies a 16-bit number A(aaa) by the value stored in the accumulator. The number in the accumulator can be up to 32 bits long. The source of the 16-bit operand can be a constant or a data value located in V-memory. Multiply Binary performs the multiplication operation on the full binary representation of the operands, which distinguishes it from the Multiply instruction (see page 5-94), which treats the operands as BCD numbers. Although the multiplication operation is performed on the underlying binary values, the native display format is hexadecimal. For that reason, you will need to load constants in hex.

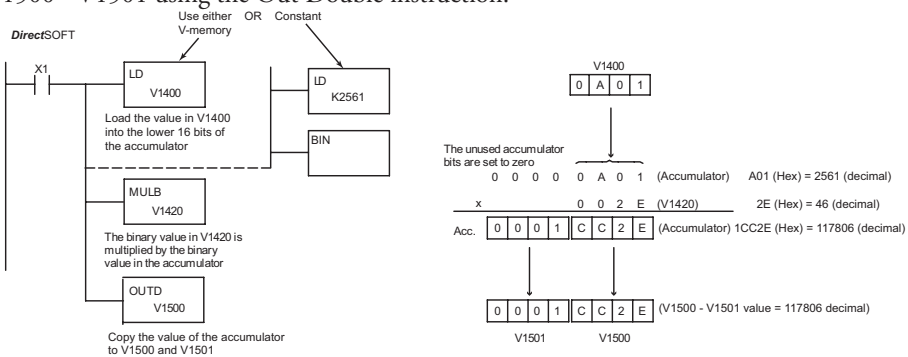
The product of the Multiply Binary operation occupies the full 32-bit accumulator and requires an Out Double to move the product to V-memory. If the value in the accumulator occupies fewer than 32 bits, leading zeros are loaded in the left-most empty bit positions.

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All (See page 3-56)	All (See page 3-57)
Pointer	P	All V mem (See page 3-56)	All V mem (See page 3-57)
Constant	K	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The binary value in V1420 is multiplied by the binary value in the accumulator using the Multiply Binary instruction. The value in the accumulator is copied to V1500 - V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

STR	→	1	ENT							
SHFT	L	D	1	4	0	0	ENT			
SHFT	M	U	L	B	→	1	4	2	0	ENT
OUT	SHFT	D	→	1	5	0	0	ENT		

Divide Binary (DIVB)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

DIVB
A aaa

The Divide Binary instruction divides a 16-bit number (Aaaa) into the value stored in the accumulator. The number in the accumulator can be up to 32 bits long. The source of the 16-bit divisor can be a constant or a data value located in V-memory. Divide Binary performs the division operation on the full binary representation of the operands, which distinguishes it from the Divide instruction (see page 5-97), which treats the operands as BCD numbers. Although the division operation is performed on the underlying binary values, the native display format is hexadecimal. For that reason you will need to load constants in hex.

At the completion of the division operation, the quotient resides in the accumulator and the remainder resides in the first stack location.

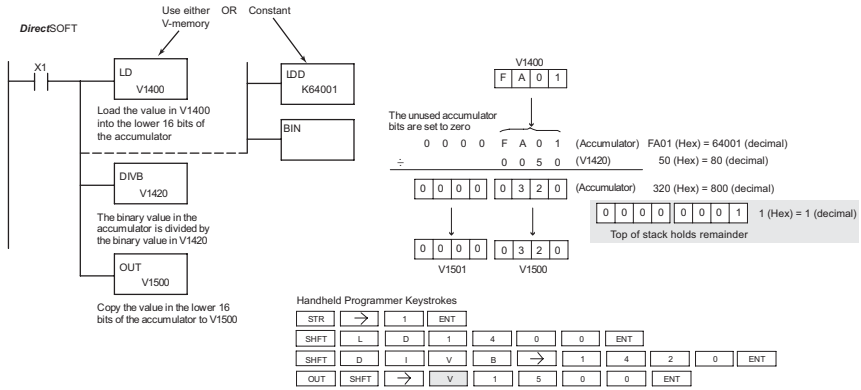
The quotient occupies the full 32-bit accumulator and requires an Out Double to move the quotient to V-memory. If the value in the accumulator occupies fewer than 32 bits, leading zeros are loaded in the left-most empty bit positions.

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All (See page 3-56)	All (See page 3-57)
Pointer	P	All V mem (See page 3-56)	All V mem (See page 3-57)
Constant	K	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

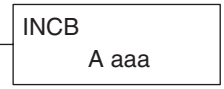
In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The binary value in the accumulator is divided by the binary value in V1420 using the Divide Binary instruction. The value in the accumulator is copied to V1500 using the Out Double instruction.



Increment Binary (INCB)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Increment Binary instruction increments a binary value in a specified V-memory location by "1" each time the instruction is executed.



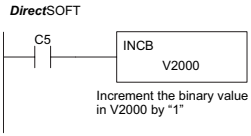
DS	Used
HPP	Used

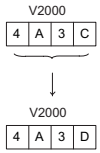
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All (See page 3-54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)
Pointer	P	-	All V-memory (See page 3-55)	All V-memory (See page 3-56)	All V-memory (See page 3-57)

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when C5 is on, the binary value in V2000 is increased by 1.





Handheld Programmer Keystrokes

$ STR	→	SHFT	C 2	F 5	ENT								
SHFT	I 8	N TMR	C 2	B 1	→	C 2	A 0	A 0	A 0	ENT			

Decrement Binary (DECB)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Decrement Binary instruction decrements a binary value a specified V-memory location by "1" each time the instructi executed.

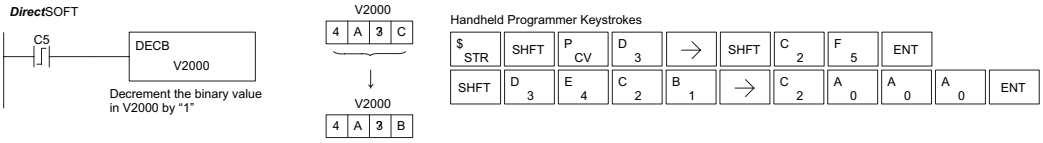
DECB
A aaa

s

Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All (See page 3-54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)
Pointer	P	-	All V-memory (See page 3-55)	All V-memory (See page 3-56)	All V-memory (See page 3-57)

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero

NOTE: *The status flags are only valid until another instruction that uses the same flag is executed.*



In the following example, when C5 is on, the value in V2000 is decreased by 1.

Add Formatted (ADDF)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

Add Formatted is a 32-bit instruction that adds the BCI value in the accumulator with the BCD value (Aaaa), which i range of discrete bits. The specified range (Kbbb) can be 1 t consecutive bits. The result resides in the accumulator.

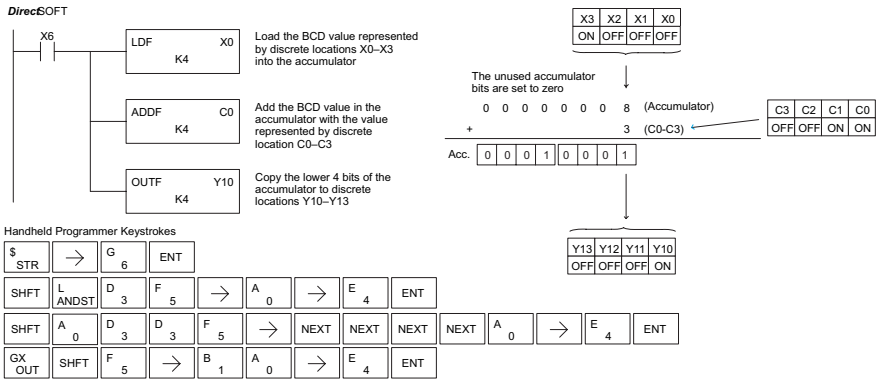
```
ADDF    A aaa
        K bbb
```

Operand Data Type		Range D2-260/D2-262	
	A	aaa	bbb
Inputs	X	0-1777	-
Outputs	Y	0-1777	-
Control Relays	C	0-3777	-
Stage Bits	S	0-1777	-
Timer Bits	T	0-377	-
Counter Bits	CT	0-377	-
Special Relays	SP	0-777	-
Global I/O	GX/GY	0-3777	-
Constant	K	-	1-32

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP66	On when the 16-bit addition instruction results in a carry
SP67	On when the 32-bit addition instruction results in a carry
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X6 is on, the value formed by discrete locations X0–X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete locations C0–C3 is added to the value in the accumulator using the Add Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10–Y13 using the Out Formatted instruction.

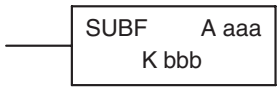


Subtract Formatted (SUBF)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

Subtract Formatted is a 32-bit instruction that subtracts the BCD value (Aaaa), which is a range of discrete bits, from the BCD value in the accumulator. The specified range (Kbbb) can be 1 to 32 consecutive bits. The result resides in the accumulator.

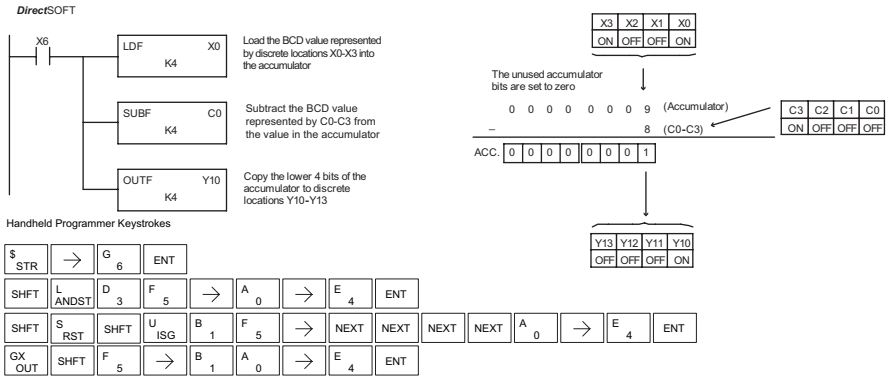


Operand Data Type		Range D2-260/D2-262	
	A	**aaa**	**bbb**
Inputs	X	0-1777	-
Outputs	Y	0-1777	-
Control Relays	C	0-3777	-
Stage Bits	S	0-1777	-
Timer Bits	T	0-377	-
Counter Bits	CT	0-377	-
Special Relays	SP	0-777	-
Global I/O	GX/GY	0-3777	-
Constant	K	-	1-32

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP64	On when the 16-bit subtraction instruction results in a borrow
SP65	On when the 32-bit subtraction instruction results in a borrow
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X6 is on, the value formed by discrete locations X0–X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete location C0–C3 is subtracted from the value in the accumulator using the Subtract Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10–Y13 using the Out Formatted instruction.

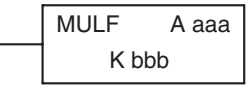


Multiply Formatted (MULF)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

Multiply Formatted is a 16-bit instruction that multiplies the BCD value in the accumulator by the BCD value (Aaaa) which is a range of discrete bits. The specified range (Kbbb) can be 1 to 16 consecutive bits. The result resides in the accumulator.

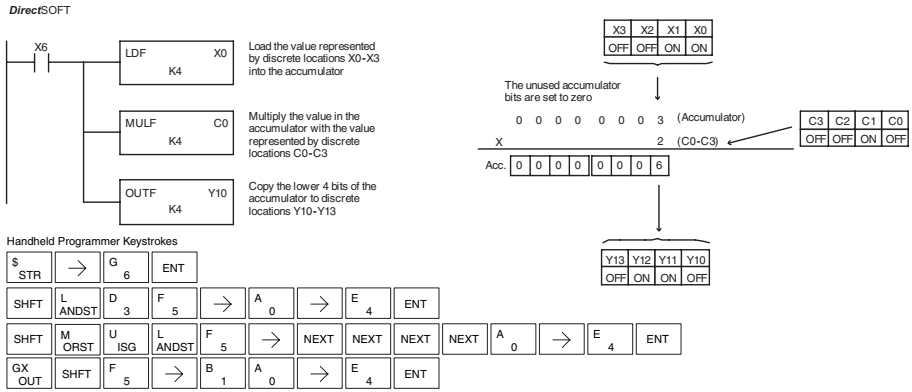


Operand Data Type		Range D2-260/D2-262	
	A	aaa	bbb
Inputs	X	0-1777	-
Outputs	Y	0-1777	-
Control Relays	C	0-3777	-
Stage Bits	S	0-1777	-
Timer Bits	T	0-377	-
Counter Bits	CT	0-377	-
Special Relays	SP	0-777	-
Global I/O	GX/GY	0-3777	-
Constant	K	-	1-16

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X6 is on, the value formed by discrete locations X0–X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete locations C0–C3 is multiplied by the value in the accumulator using the Multiply Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10–Y13 using the Out Formatted instruction.



Divide Formatted (DIVF)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

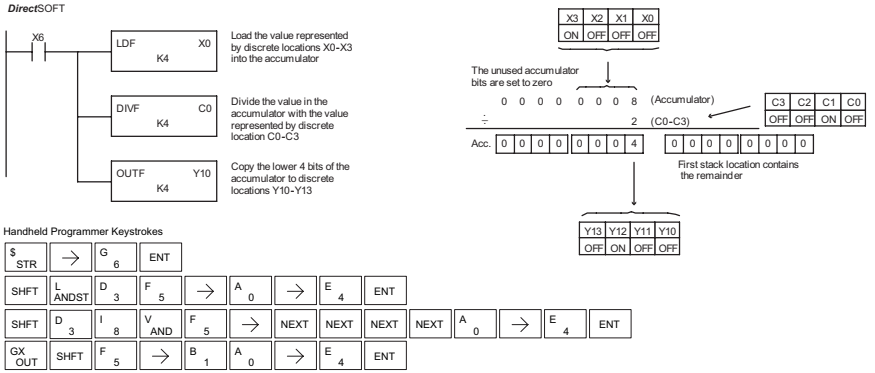
Divide Formatted is a 16-bit instruction that divides the BCD value in the accumuator by the BCD value (Aaaa), a range of discrete bits. The specified range (Kbbb) can be 1 to 16 consecutive bits. The first part of the quotient resides in the accumulator and the remainder resides in the first stack location.

Operand Data Type		Range D2-260/D2-262	
	A	aaa	bbb
Inputs	X	0-1777	-
Outputs	Y	0-1777	-
Control Relays	C	0-3777	-
Stage Bits	S	0-1777	-
Timer Bits	T	0-377	-
Counter Bits	CT	0-377	-
Special Relays	SP	0-777	-
Global I/O	GX/GY	0-3777	-
Constant	K	-	1-16

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

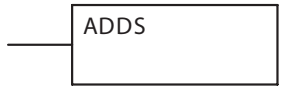
In the following example, when X6 is on, the value formed by discrete locations X0–X3 is loaded into the accumulator using the Load Formatted instruction. The value in the accumulator is divided by the value formed by discrete location C0–C3 using the Divide Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10–Y13 using the Out Formatted instruction.



Add Top of Stack (ADDS)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

Add Top of Stack is a 32-bit instruction that adds the BCD value in the accumulator with the BCD value in the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack values are moved up one level.

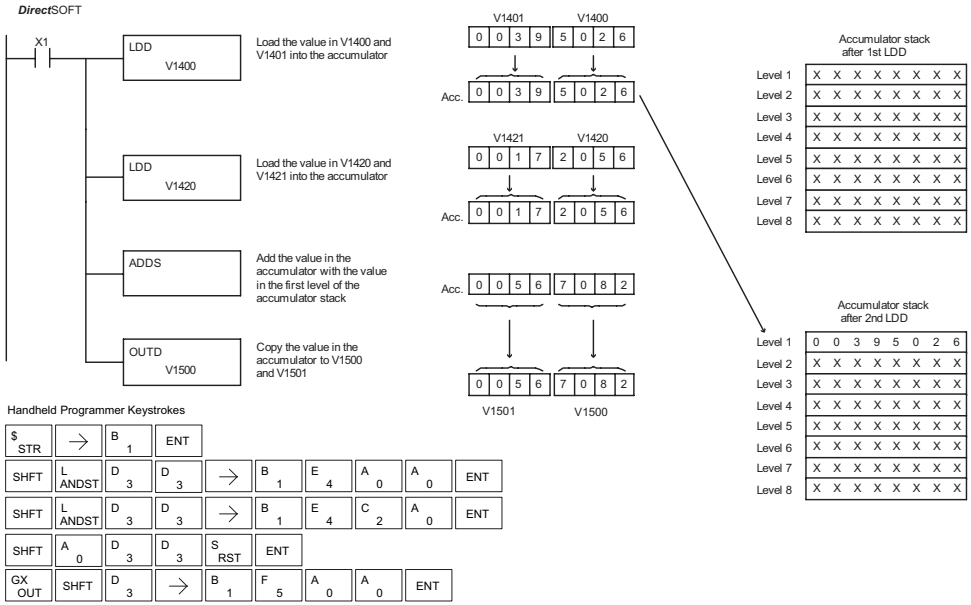


Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP66	On when the 16-bit addition instruction results in a carr
SP67	On when the 32-bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negativ.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

DS	Used
HPP	Used

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The value in the first level of the accumulator stack is added with the value in the accumulator using the Add Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



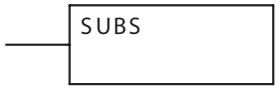
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	C 2	A 0	ENT
SHFT	A 0	D 3	D 3	S RST	ENT				
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

Subtract Top of Stack (SUBS)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

Subtract Top of Stack is a 32-bit instruction that subtracts the BCD value in the first level of the accumulator stack from the BCD value in the accumulator. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack values are moved up one level.

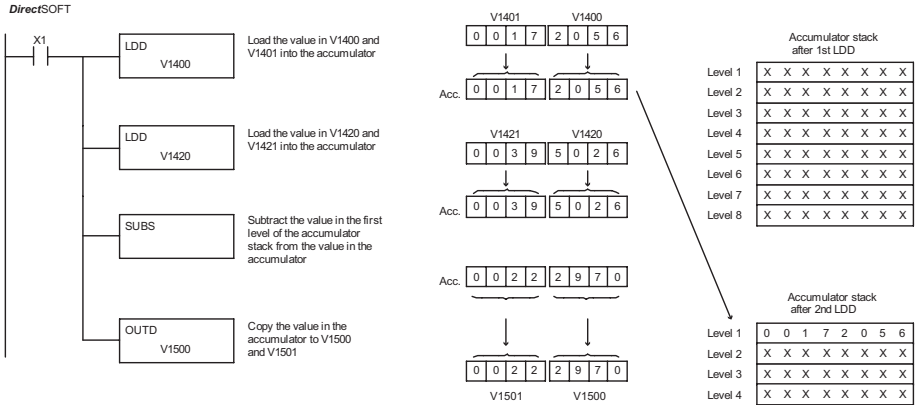


Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP64	On when the 16-bit subtraction instruction results in a borrow
SP65	On when the 32-bit subtraction instruction results in a borrow
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

DS	Used
HPP	Used

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded into the accumulator onto the accumulator stack. The BCD value in the first level of the accumulator stack is subtracted from the BCD value in the accumulator using the Subtract Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



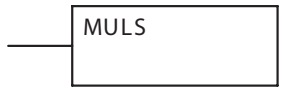
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT											
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT					
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	C 2	A 0	ENT					
SHFT	S RST	SHFT	U ISG	B 1	S RST	ENT								
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT						

Multiply Top of Stack (MULS)

☒ 230
☒ 240
☒ 250-1
☑ **260**
☑ **262**

Multiply Top of Stack is a 16-bit instruction that multiplies a 4-digit BCD value in the first level of the accumulator stack by a 4-digit BCD value in the accumulator. The result resides in the accumulator. The value in the first level of the accumulator stack is removed, and all stack values are moved up one level.

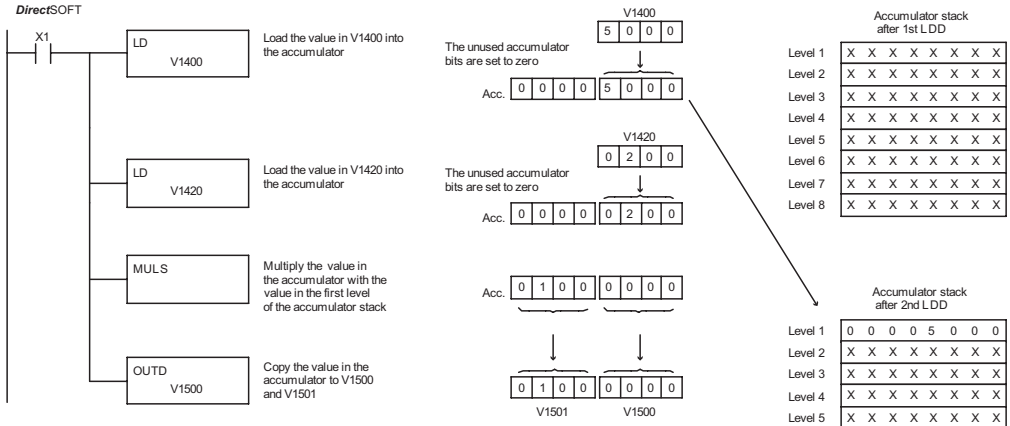


Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

DS	Used
HPP	Used

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The value in V1420 is loaded into the accumulator using the Load instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The BCD value in the first level of the accumulator stack is multiplied by the BCD value in the accumulator using the Multiply Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	→	B 1	E 4	C 2	A 0	ENT
SHFT	M ORST	U ISG	L ANDST	S RST	ENT			
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT

Divide by Top of Stack (DIVS)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

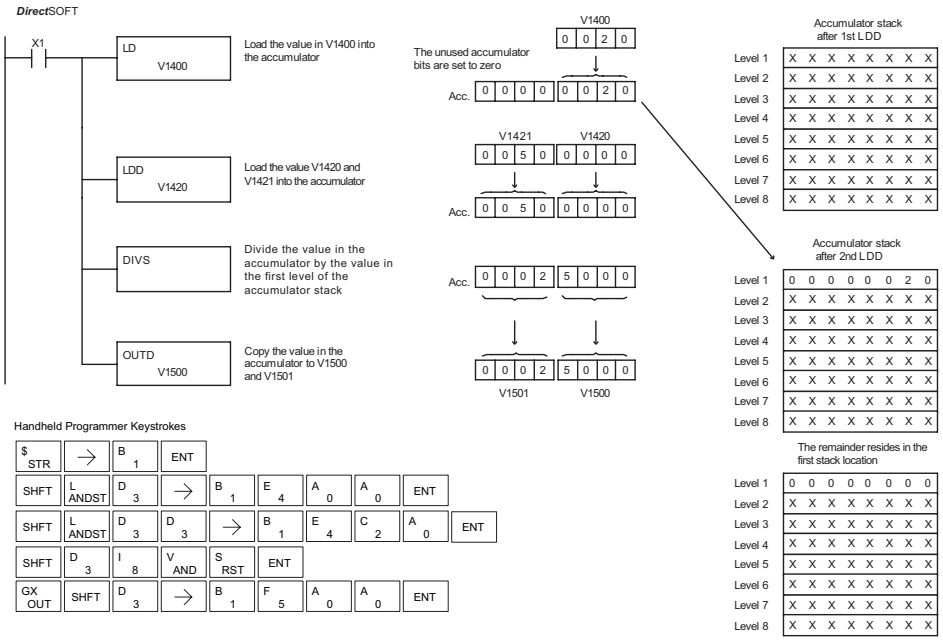
Divide Top of Stack is a 32-bit instruction that divides the 8-digit BCD value in the accumulator by a 4-digit BCD value in the first level of the accumulator stack. The result resides in the accumulator and the remainder resides in the first level of the accumulator stack.

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered

NOTE: *Status flags are valid only until another instruction uses the same flag.*

DS	Used
HPP	Used

In the following example, when X1 is on, the Load instruction loads the value in V1400 into the accumulator. The value in V1420 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The BCD value in the accumulator is divided by the BCD value in the first level of the accumulator stack using the Divide Stack instruction. The Out Double instruction copies the value in the accumulator to V1500 and V1501.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	C 2	A 0	ENT
SHFT	D 3	I 8	V AND	S RST	ENT			
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT

Add Binary Top of Stack (ADDBS)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

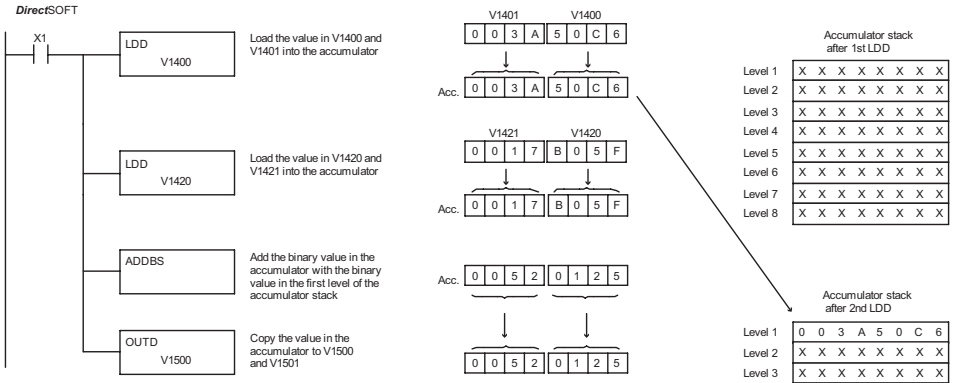
Add Binary Top of Stack instruction is a 32-bit instruction that adds the binary value in the accumulator with the binary value in the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed, and all stack values are moved up one level.

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP66	On when the 16-bit addition instruction results in a carry
SP67	On when the 32-bit addition instruction results in a carry
SP70	On anytime the value in the accumulator is negative
SP73	On when a signed addition or subtraction results in a incorrect sign bit

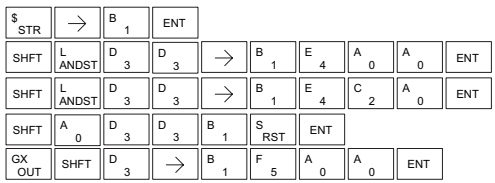
NOTE: *Status flags are valid only until another instruction uses the same flag.*

DS	Used
HPP	Used

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The binary value in the first level of the accumulator stack is added with the binary value in the accumulator using the Add Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes



Subtract Binary Top of Stack (SUBBS)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

Subtract Binary Top of Stack is a 32-bit instruction that subtracts the binary value in the first level of the accumulator stack from the binary value in the accumulator. The result resides in the accumulator. The value in the first level of the accumulator stack is removed, and all stack locations are moved up one level.

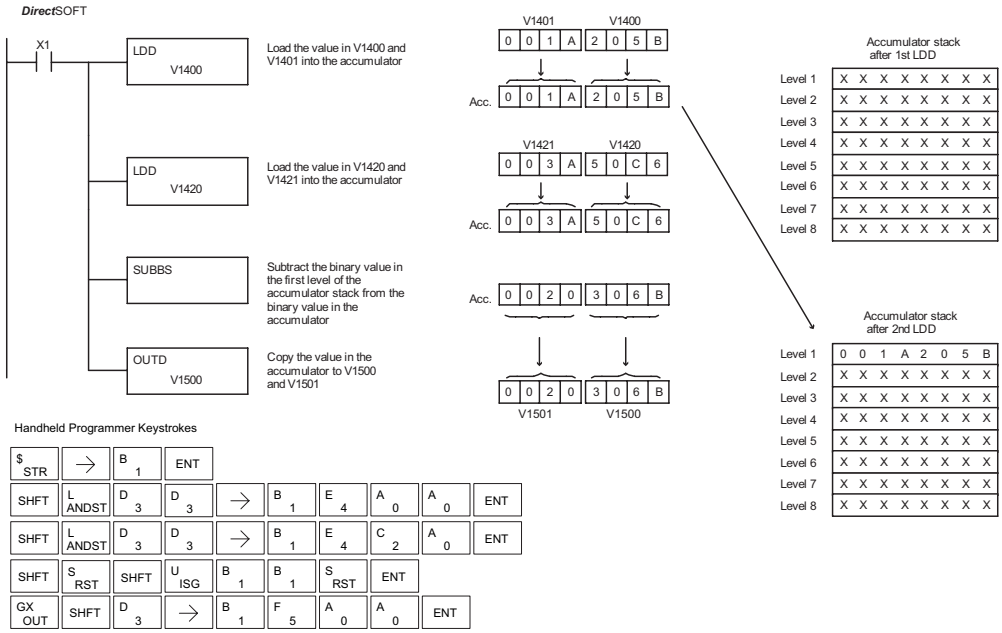
SUBBS

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP64	On when the 16-bit subtraction instruction results in a borrow
SP65	On when the 32-bit subtraction instruction results in a borrow
SP70	On anytime the value in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

DS	Used
HPP	Used

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The binary value in the first level of the accumulator stack is subtracted from the binary value in the accumulator using the Subtract Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



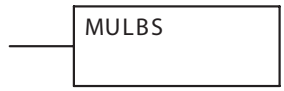
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	C 2	A 0	ENT
SHFT	S RST	SHFT	U ISG	B 1	B 1	S RST	ENT		
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

Multiply Binary Top of Stack (MULBS)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

Multiply Binary Top of Stack is a 16-bit instruction that multiplies the 16-bit binary value in the first level of the accumulator stack by the 16-bit binary value in the accumulator. The result resides in the accumulator and can be 32 bits (8 digits maximum). The value in the first level of the accumulator stack is removed, and all stack locations are moved up one level.

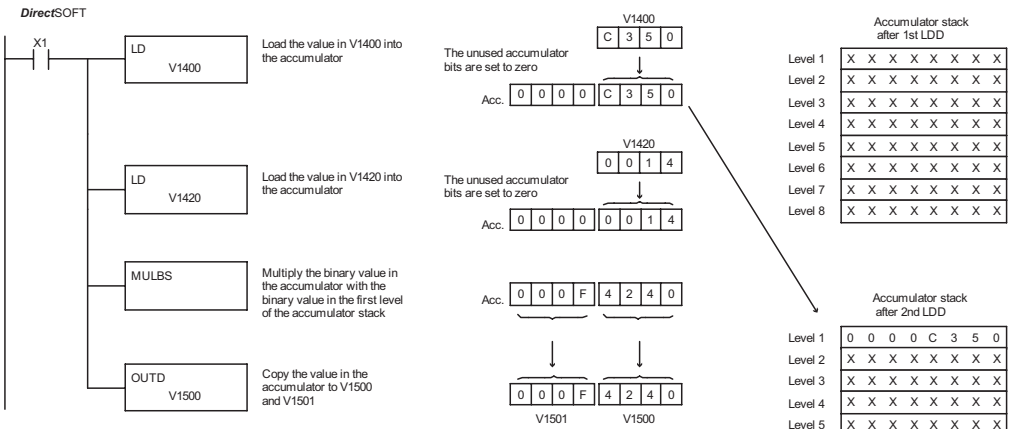


Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

DS	Used
HPP	Used

In the following example, when X1 is on, the Load instruction moves the value in V1400 into the accumulator. The value in V1420 is loaded into the accumulator using the Load instruction, pushing the value previously loaded in the accumulator onto the stack. The binary value in the accumulator stack's first level is multiplied by the binary value in the accumulator using the Multiply Binary Stack instruction. The Out Double instruction copies the value in the accumulator to V1500 and V1501.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	→	B 1	E 4	C 2	A 0	ENT
SHFT	M ORST	U ISG	L ANDST	B 1	S RST	ENT		
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT

Divide Binary by Top of Stack (DIVBS)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

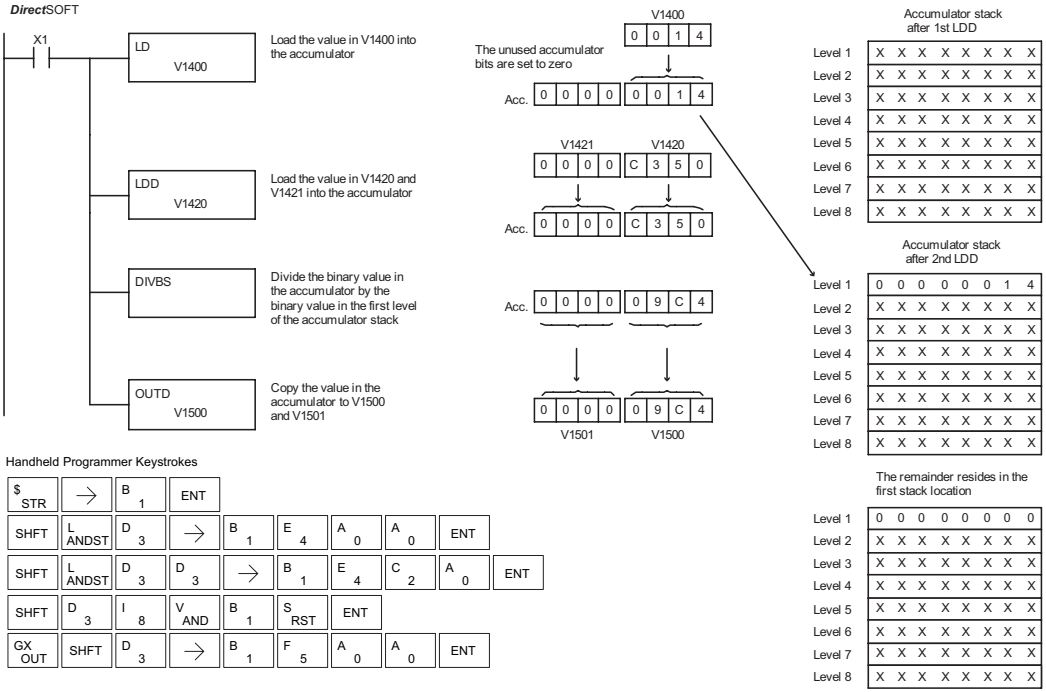
Divide Binary Top of Stack is a 32-bit instruction that divides the 32-bit binary value in the accumulator by the 16-bit binary value in the first level of the accumulator stack. The result resides in the accumulator. and the remainder resides in the first level of the accumulator stack.

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

DS	Used
HPP	Used

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The binary value in the accumulator is divided by the binary value in the first level of the accumulator stack using the Divide Binary Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	C 2	A 0	ENT
SHFT	D 3	I 8	V AND	B 1	S RST	ENT			
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT

Transcendental Functions (D2-260 and D2-262 only)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

The D2-260 and D2-262 CPUs feature special numerical functions to complement the real number capability. The transcendental functions include the trigonometric sine, cosine, and tangent, and also their inverses (arc sine, arc cosine, and arc tangent). The square root function is also grouped with these other functions.

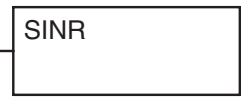
The transcendental math instructions operate on a real number in the accumulator (it cannot be BCD or binary). The real number result resides in the accumulator. The square root function operates on the full range of positive real numbers. The sine, cosine and tangent functions require numbers expressed in radians. You can work with angles expressed in degrees by first converting them to radians with the Radian (RADR) instruction, then performing the trig function. All transcendental functions utilize the following flag bits.

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP72	On anytime the value in the accumulator is a valid floating point number
SP73	On when a signed addition or subtraction results in a incorrect sign bit
SP75	On when a real number instruction is executed and a non-real number was encountered

Math Function	Range of Argument
SP53	On when the value of the operand is larger than the accumulator can work with

Sine Real (SINR)

The Sine Real instruction takes the sine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.



Cosine Real (COSR)

The Cosine Real instruction takes the cosine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

Tangent Real (TANR)

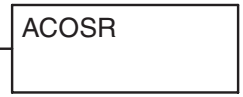
The Tangent Real instruction takes the tangent of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

Arc Sine Real (ASINR)

The Arc Sine Real instruction takes the inverse sine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

Arc Cosine Real (ACOSR)

The Arc Cosine Real instruction takes the inverse cosine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.



Arc Tangent Real (ATANR)

The Arc Tangent Real instruction takes the inverse tangent of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

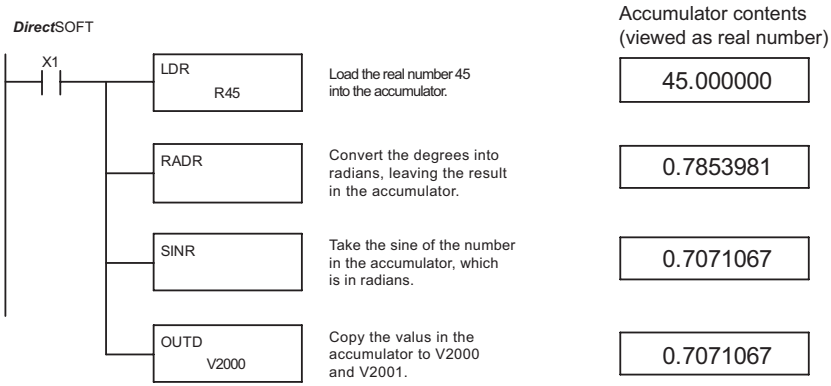
Square Root Real (SQRTR)

The Square Root Real instruction takes the square root of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

NOTE: *The square root function can be useful in several situations. However, if you are trying to do the square-root extract function for an orifice flow meter measurement as the PV to a PID loop, note that the PID loop already has the square-root extract function built in.*

DS	Used
HPP	N/A

The following example takes the **sine** of 45 degrees. Since these transcendental functions operate only on real numbers, we do a LDR (Load Real) 45. The trig functions operate only in radians, so we must convert the degrees to radians by using the RADR command. After using the SINR (Sine Real) instruction, we use an OUTD (Out Double) instruction to move the result from the accumulator to V-memory. The result is 32-bits wide, requiring the Out Double to move it.



NOTE: *The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use* ***Direct****SOFT for entering real numbers, using the LDR (Load Real) instruction.*

Bit Operation Instructions

Sum (SUM)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

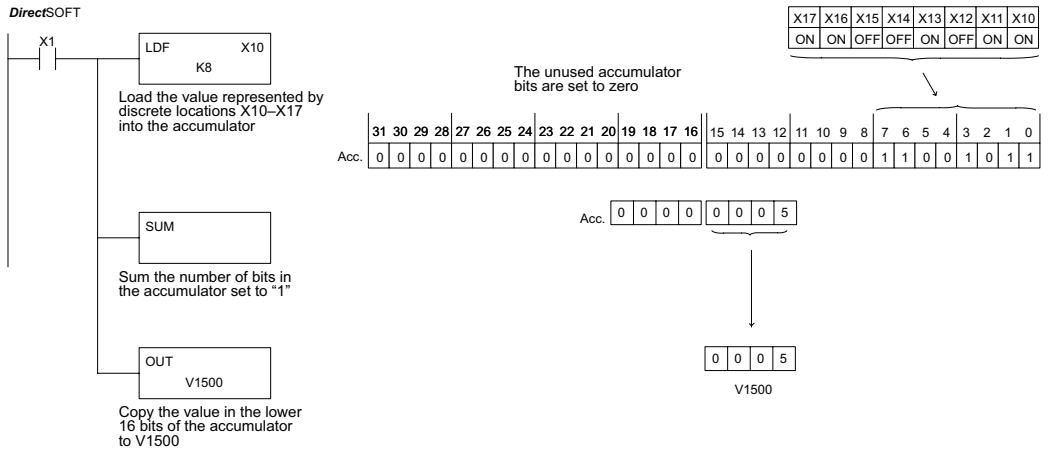
DS	Used
HPP	Used

The Sum instruction counts the number of bits that are set to "1" in the accumulator. The HEX result resides in the accumulator.

SUM

Math Function	Range of Argument
SP63	On when the result of the instruction causes the value in the accumulator to be zero

In the following example, when X1 is on, the value formed by discrete locations X10–X17 is loaded into the accumulator using the Load Formatted instruction. The number of bits in the accumulator set to "1" is counted using the Sum instruction. The value in the accumulator is copied to V1500 using the Out instruction.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	F 5	→	B 1	A 0	→	I 8	ENT
SHFT	S RST	SHFT	U ISG	M ORST	→	ENT			
GX OUT	→	PREV	PREV	PREV	B 1	F 5	A 0	A 0	ENT

Shift Left (SHFL)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

Shift Left is a 32-bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the left. The vacant positions are filled with zeros, and the bits shifted out of the accumulator are lost.

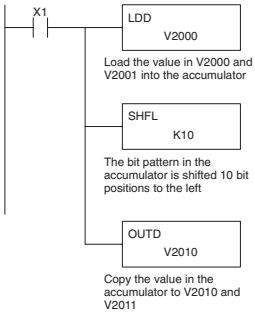
SHFL
A aaa

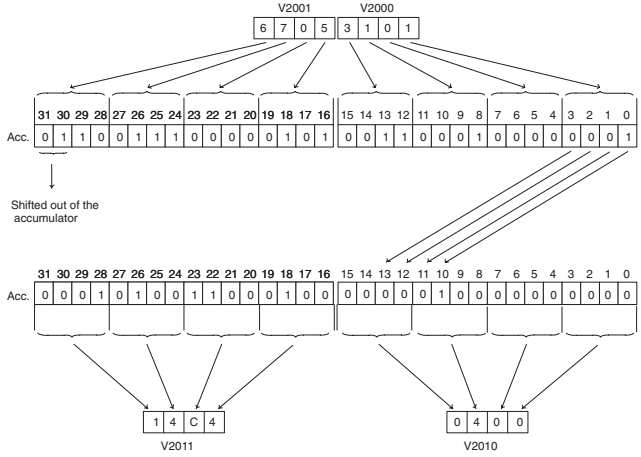
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All (See page 3-54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)
Constant	K	1-32	1-32	1-32	1-32

DS	Used
HPP	Used

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is shifted 10 bits to the left using the Shift Left instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.

Direct SOFT





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	S RST	SHFT	H 7	F 5	L ANDST	→	B 1	A 0	ENT
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

Shift Right (SHFR)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

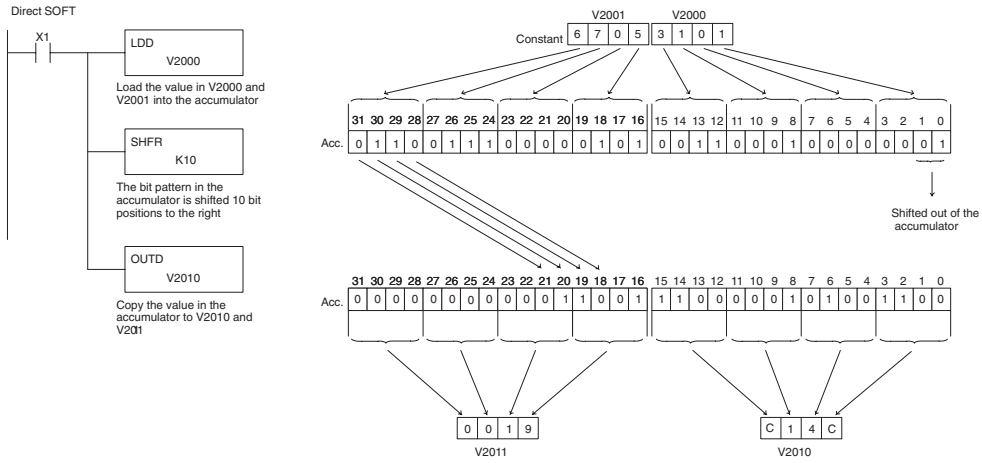
Shift Right is a 32-bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the right. The vacant positions are filled with zeros, and the bits shifted out of the accumulator are lost.

SHFR
A aaa

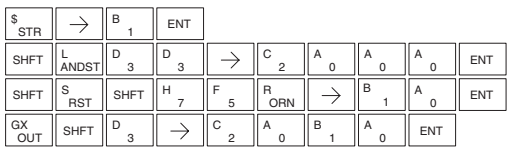
Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa	aaa
V-memory	V	All (See page 3-54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)
Constant	K	1-32	1-32	1-32	1-32

DS	Used
HPP	Used

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is shifted 10 bits to the right using the Shift Right instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes



Rotate Left (ROTL)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

Rotate Left is a 32-bit instruction that rotates the bits in the accumulator a specified number (Aaaa) of places to the left.

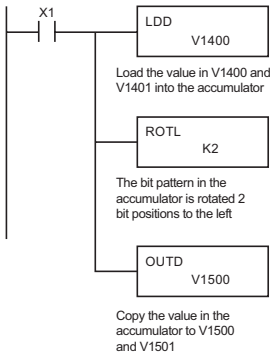
ROTL
A aaa

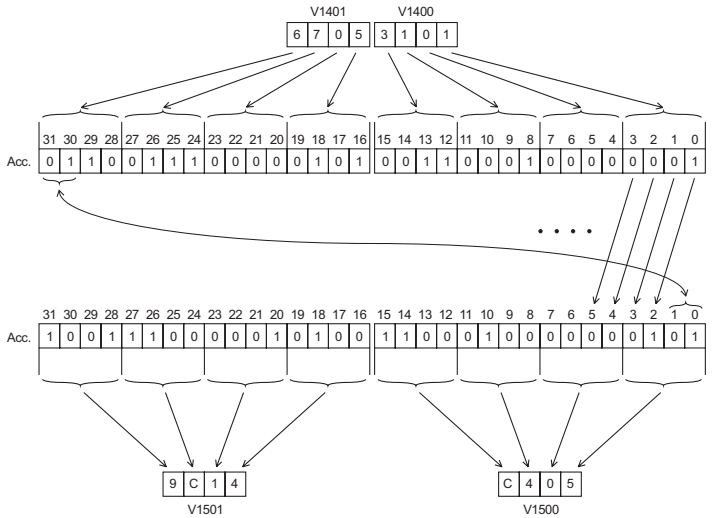
Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All (See page 3-56)	All (See page 3-57)
Constant	K	1-32	1-32

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is rotated 2 bit positions to the left using the Rotate Left instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

DS	Used
HPP	Used

*Direct*SOFT





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	R ORN	O INST#	T MLR	L ANDST	→	C 2	ENT		
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

Rotate Right (ROTR)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

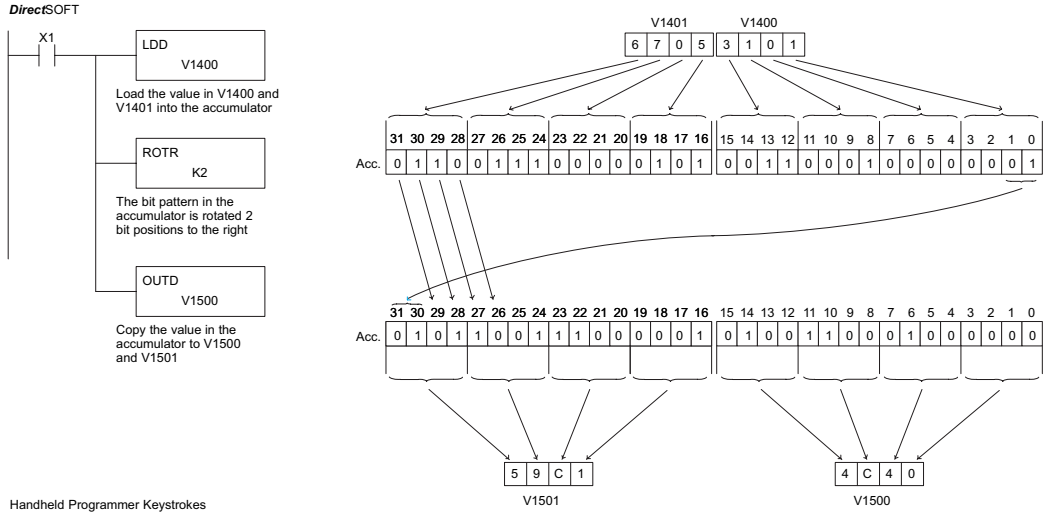
Rotate Right is a 32-bit instruction that rotates the bits in the accumulator a specified number (Aaaa) of places to the right.

ROTR
A aaa

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
V-memory	V	All (See page 3-56)	All (See page 3-57)
Constant	K	1-32	1-32

DS	Used
HPP	Used

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is rotated 2 bit positions to the right using the Rotate Right instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	R ORN	O INST#	T MLR	R ORN	→	C 2	ENT		
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

Encode (ENCO)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

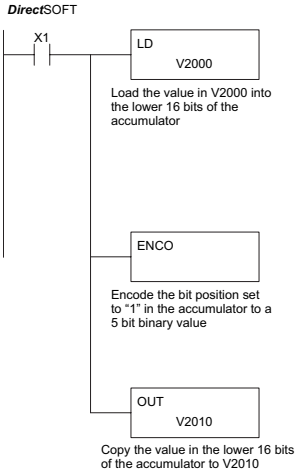
DS	Used
HPP	Used

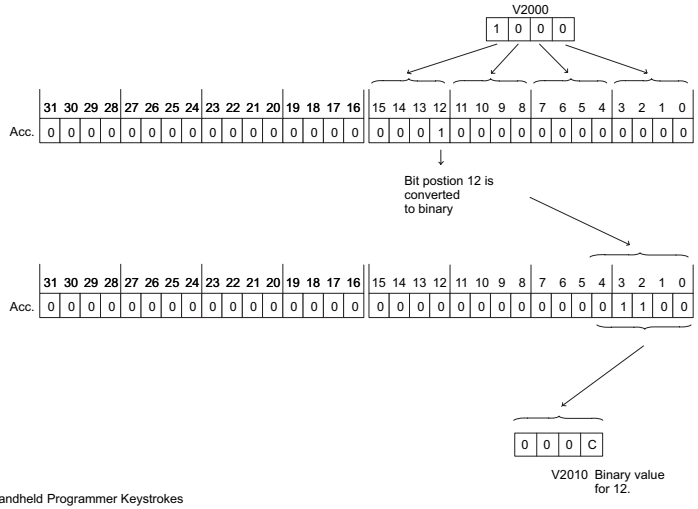
The Encode instruction encodes the bit position in the accumulator having a value of 1, and returns the appropriate binary representation. If the most significant bit is set to 1 (Bit 31), the Encode instruction would place the value HEX 1F (decimal 31) in the accumulator. If the value to be encoded is 0000 or 0001, the instruction will place a zero in the accumulator. If the value to be encoded has more than one bit position set to a "1", the least significant "1" will be encoded and SP53 will be set on.

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with

NOTE: *The status flags are only valid until another instruction that uses the same flags is executed.*

In the following example, when X1 is on, The value in V2000 is loaded into the accumulator using the Load instruction. The bit position set to a "1" in the accumulator is encoded to the corresponding 5-bit binary value using the Encode instruction. The value in the lower 16 bits of the accumulator is copied to V2010 using the Out instruction.





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	E 4	N TMR	C 2	O INST#	ENT			
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

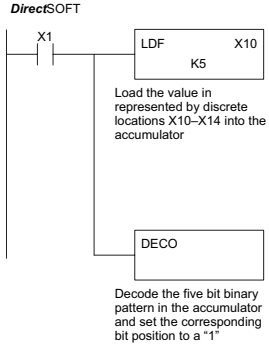
Decode (DECO)

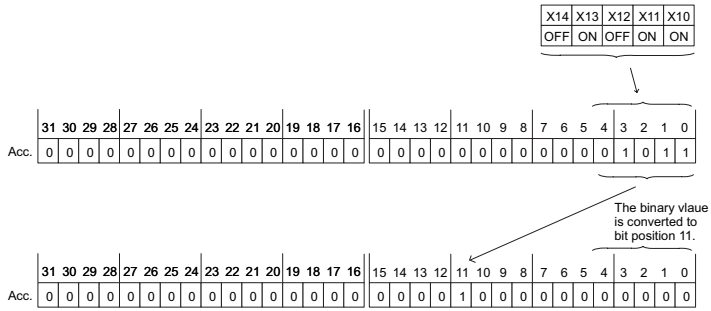
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Decode instruction decodes a 5-bit binary value of 0 to 31 (0 to 1F HEX) in the accumulator by setting the appropriate bit position to a 1. If the accumulator contains the value F (HEX), bit 15 will be set in the accumulator. If the value to be decoded is greater than 31, the number is divided by 32 until the value is less than 32 and then the value is decoded.

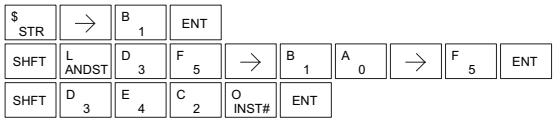
DS	Used
HPP	Used

In the following example, when X1 is on, the value formed by discrete locations X10–X14 is loaded into the accumulator using the Load Formatted instruction. The 5-bit binary pattern in the accumulator is decoded by setting the corresponding bit position to a "1" using the Decode instruction.





Handheld Programmer Keystrokes



Number Conversion Instructions (Accumulator)

Binary (BIN)

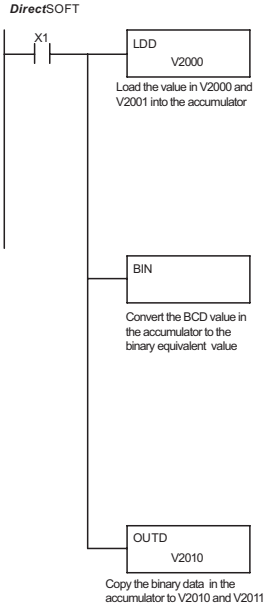
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

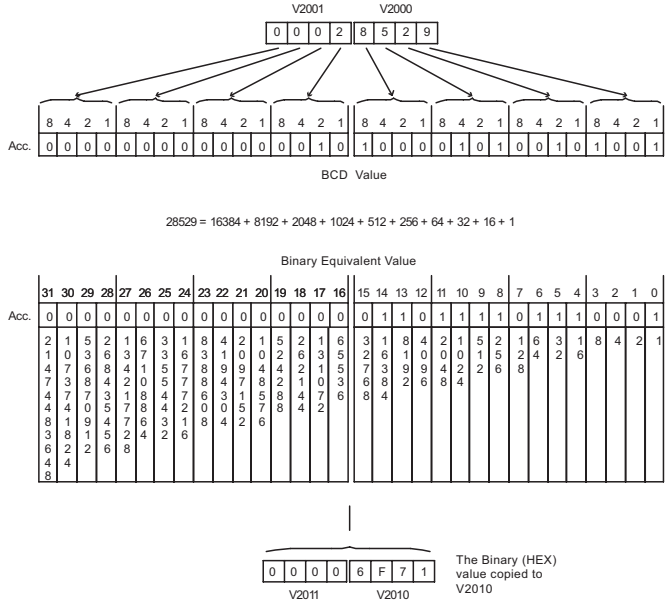
DS	Used
HPP	Used

BIN

The Binary instruction converts a BCD value in the accumulator to the equivalent binary value. The result resides in the accumulator.

In the following example, when X1 is on, the value in V2000 and V2001 is loaded into the accumulator using the Load Double instruction. The BCD value in the accumulator is converted to the binary (HEX) equivalent using the BIN instruction. The binary value in the accumulator is copied to V2010 and V2011 using the Out Double instruction. (The Handheld Programmer will display the binary value in V2010 and V2011 as a HEX value.)





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	B 1	I 8	N TMR	ENT					
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

Binary Coded Decimal (BCD)

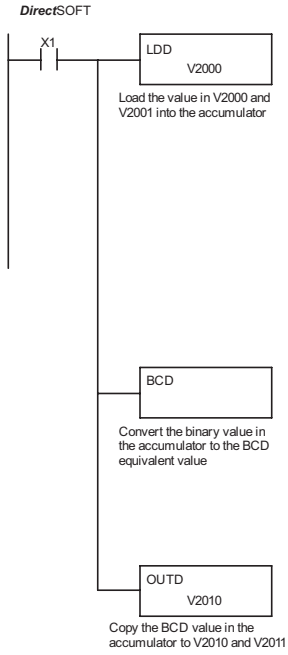
BCD

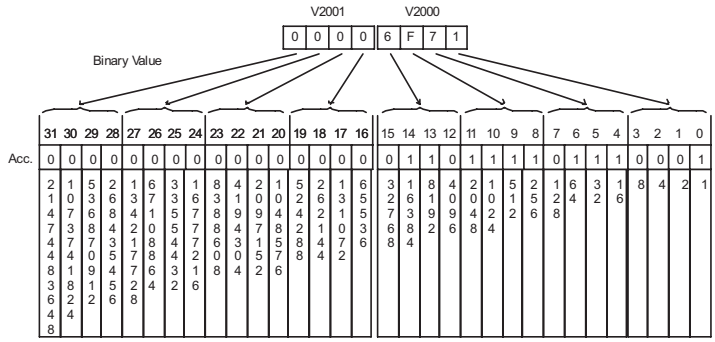
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Binary Coded Decimal instruction converts a binary value in the accumulator to the equivalent BCD value. The result resides in the accumulator.

In the following example, when X1 is on, the binary (HEX) value in V2000 and V2001 is loaded into the accumulator using the Load Double instruction. The binary value in the accumulator is converted to the BCD equivalent value using the BCD instruction. The BCD value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.





Handheld Programmer Keystrokes

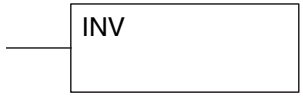
$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	B 1	C 2	D 3	ENT					
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

Invert (INV)

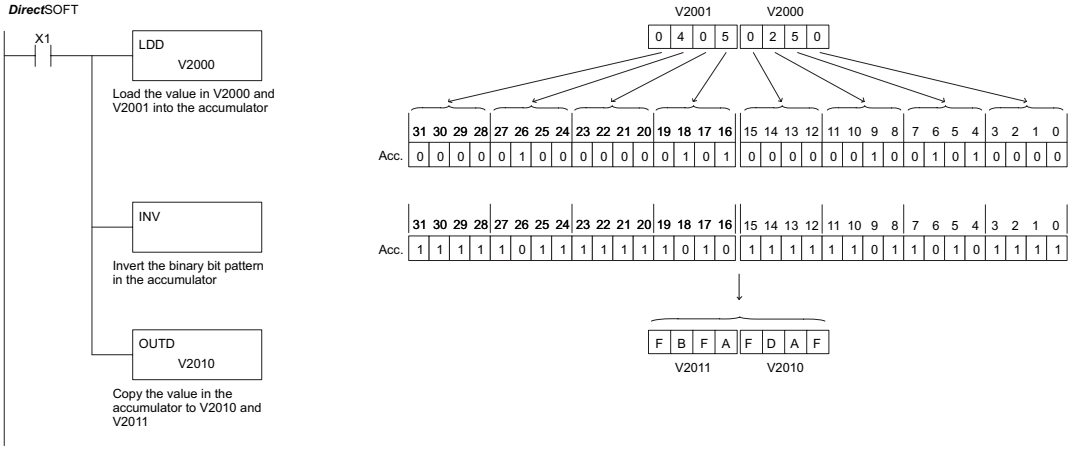
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Invert instruction inverts or takes the one's complement of the 32-bit value in the accumulator. The result resides in the accumulator.



In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is inverted using the Invert instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	I 8	N TMR	V AND	ENT					
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

Ten's Complement (BCDCPL)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

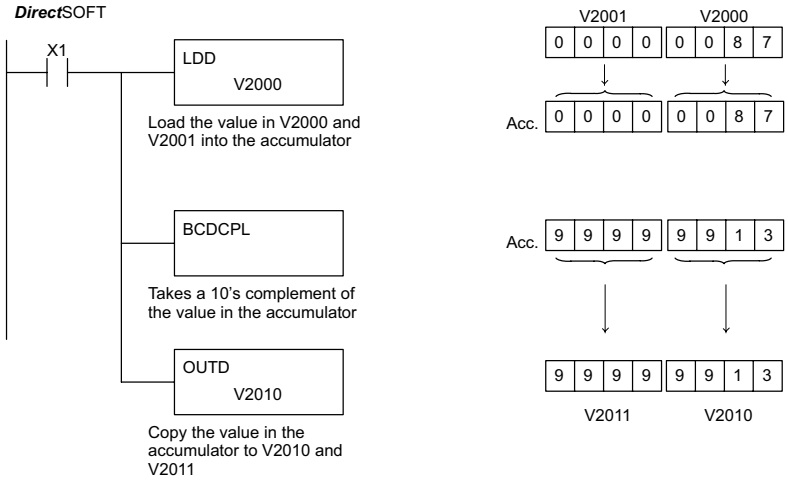
DS	Used
HPP	Used

BCDCPL

The Ten's Complement instruction takes the 10's complement (BCD) of the 8-digit accumulator. The result resides in the accumulator. The calculation for this instruction is :

100000000
— accumulator value
10's complement value

In the following example when X1 is on, the value in V2000 and V2001 is loaded into the accumulator. The 10's complement is taken for the 8-digit accumulator using the Ten's Complement instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	B 1	C 2	D 3	C 2	P CV	L ANDST	ENT		
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

Binary to Real Conversion (BTOR)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

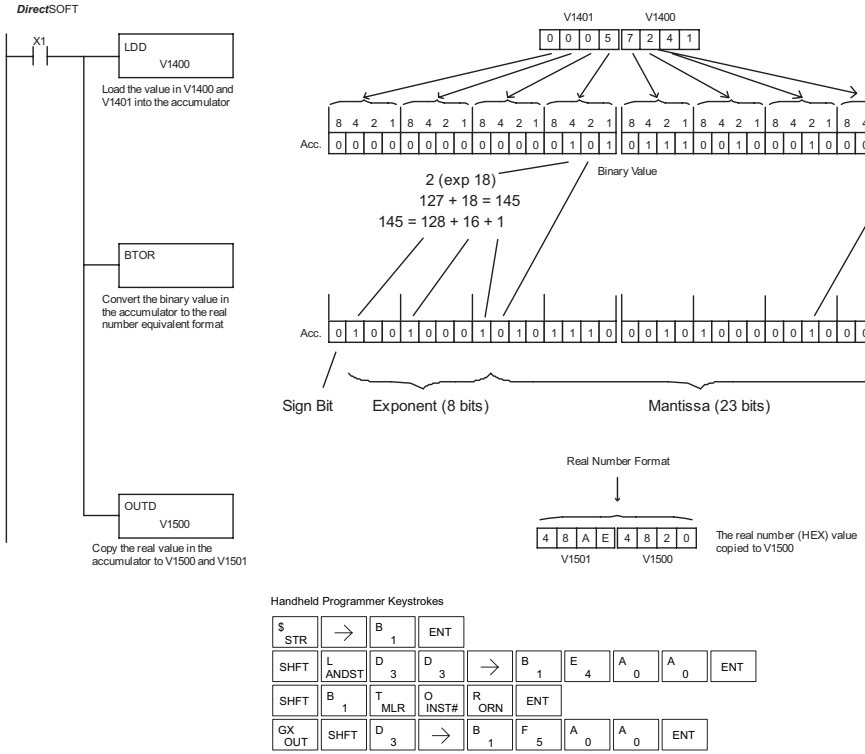
The Binary-to-Real instruction converts a binary value in the accumulator to its equivalent real number (floating point) format. The result resides in the accumulator. Both the binary and the real number may use all 32 bits of the accumulator.

NOTE: *This instruction only works with unsigned* ***binary, or decimal*** *values. It will not work with signed decimal values.*

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative

In the following example, when X1 is on, the value in V1400 and V1401 is loaded into the accumulator using the Load Double instruction. The BTOR instruction converts the binary value in the accumulator the equivalent real number format. The binary weight of the MSB is converted to the real number exponent by adding it to 127 (decimal). Then the remaining bits are copied to the mantissa as shown. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction. The Handheld Programmer would display the binary value in V1500 and V1501 as a HEX value.

DS	Used
HPP	Used



Real to Binary Conversion (RTOB)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

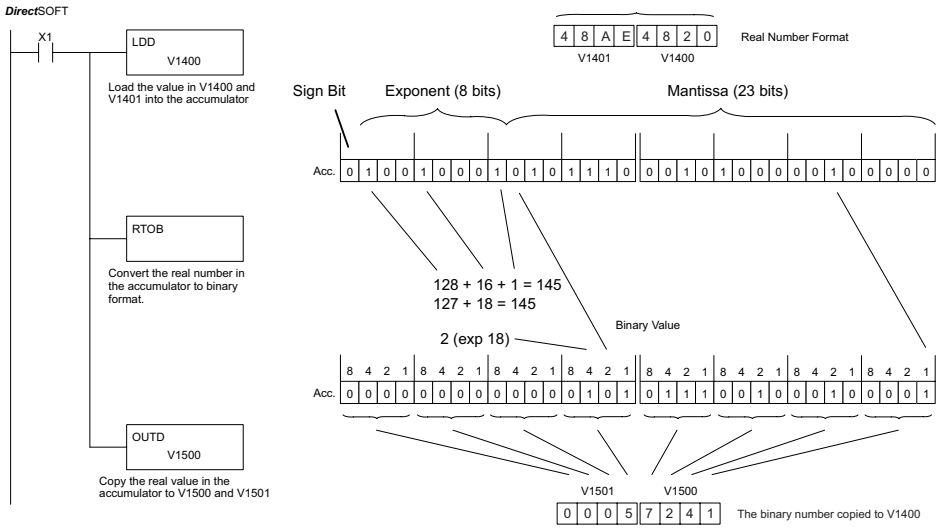
The Real-to-Binary instruction converts the real number in the accumulator to a binary value. The result resides in the accumulator. Both the binary and the real number may use all 32 bits of the accumulator.

NOTE 1: *The decimal portion of the result will be rounded down (14.1 to 14 or - 14.1 to -15).*
NOTE 2*: If the real number is negative, it becomes a signed decimal value.*

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP72	On anytime the value in the accumulator is a valid floating point number
SP73	On when a signed addition or subtraction results in an incorrect sign bit.
SP75	On when a number cannot be converted to binary

DS	Used
HPP	Used

In the following example, when X1 is on, the value in V1400 and V1401 is loaded into the accumulator using the Load Double instruction. The RTOB instruction converts the real value in the accumulator the equivalent binary number format. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction. The Handheld Programmer would display the binary value in V1500 and V1501 as a HEX value.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
SHFT	R ORN	T MLR	O INST#	B 1	ENT				
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

Radian Real Conversion (RADR)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

The Radian Real Conversion instruction converts the real degree value stored in the accumulator to the equivalent real number in radians. The result resides in the accumulator.

RADR

Degree Real Conversion (DEGR)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

The Degree Real instruction converts the degree real radian value stored in the accumulator to the equivalent real number in degrees. The result resides in the accumulator.

DEGR

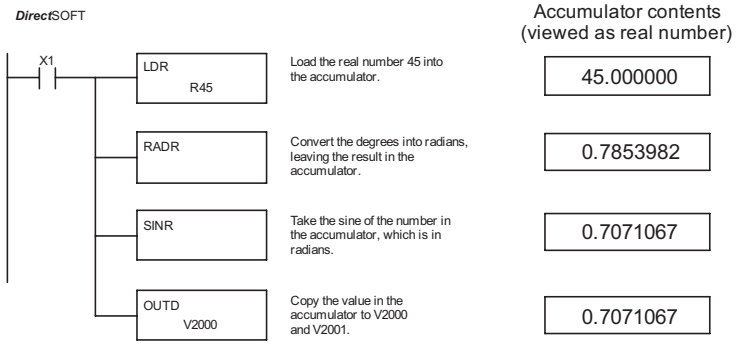
The two instructions described above convert real numbers in the accumulator from degree format to radian format, and visa-versa. In degree format, a circle contains 360 degrees. In radian format, a circle contains 2 Π. These convert between both positive and negative real numbers, and for angles greater than a full circle. These functions are very useful when combined with the transcendental trigonometric functions (see the section on math instructions).

DS	Used
HPP	N/A

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero
SP70	On anytime the value in the accumulator is negative
SP71	On anytime the V-memory specified by a pointer (P) is not valid
SP72	On anytime the value in the accumulator is a valid floating point number
SP74	On anytime a floating point math operation results in an underflow error
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered

NOTE: *The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use* ***Direct****SOFT for entering real numbers, using the LDR (Load Real) instruction.*

The following example takes the sine of 45 degrees. Since transcendental functions operate only on real numbers, we do a LDR (Load Real) 45. The trig functions operate only in radians, so we must convert the degrees to radians by using the RADR command. After using the SINR (Sine Real) instruction, we use an OUTD (Out Double) instruction to move the result from the accumulator to V-memory. The result is 32-bits wide, requiring the Out Double to move it.

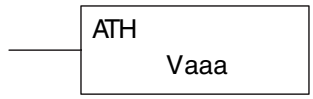


ASCII to HEX (ATH)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

The ASCII TO HEX instruction converts a table of ASCII values to a specified table of HEX values. ASCII values are two digits and their HEX equivalents are one digit.



This means an ASCII table of four V-memory locations would only require two V-memory locations for the equivalent HEX table. The function parameters are loaded into the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program an ASCII to HEX table function. The example on the following page shows a program for the ASCII to HEX table function.

Step 1: Load the number of V-memory locations for the ASCII table into the first level of the accumulator stack.

Step 2: Load the starting V-memory location for the ASCII table into the accumulator. This parameter must be a HEX value.

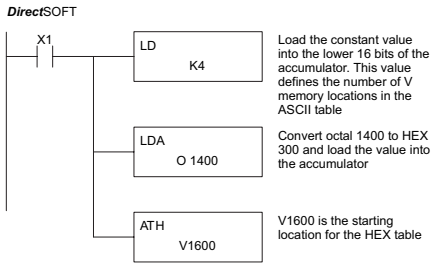
Step 3: Specify the starting V-memory location (Vaaa) for the HEX table in the ATH instruction.

Helpful hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
		aaa	aaa
V-memory	V	All (See page 3-56)	All (See page 3-57)

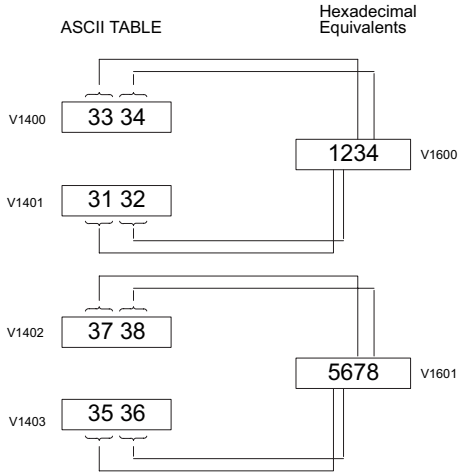
In the example on the following page, when X1 is ON, the constant (K4) is loaded into the accumulator using the Load instruction and will be placed in the first level of the accumulator stack when the next Load instruction is executed. The starting location for the ASCII table (V1400) is loaded into the accumulator using the Load Address instruction. The starting location for the HEX table (V1600) is specified in the ASCII to HEX instruction. The table below lists valid ASCII values for ATH conversion.

ASCII Values Valid for ATH Conversion			
ASCII	Hex Value	ASCII Value	Hex Value
30	0	38	8
31	1	39	9
32	2	41	A
33	3	42	B
34	4	43	C
35	5	44	D
36	6	45	E
37	7	46	F



Handheld Programmer Keystrokes

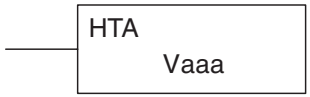
$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	PREV	E 4	ENT			
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT
SHFT	A 0	T MLR	H 7	→	B 1	G 6	A 0	A 0	ENT



HEX to ASCII (HTA)

- ☒ 230
- ☒ 240
- ☑ 250-1
- ☑ 260
- ☑ 262

The HEX to ASCII instruction converts a table of HEX values to a specified table of ASCII values. HEX values are one digit and their ASCII equivalents are two digits.



This means a HEX table of two V-memory locations would require four V-memory locations for the equivalent ASCII table. The function parameters are loaded into the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program a HEX to ASCII table function. The example on the following page shows a program for the HEX to ASCII table function.

DS	Used
HPP	N/A

Step 1: Load the number of V-memory locations in the HEX table into the first level of the accumulator stack.

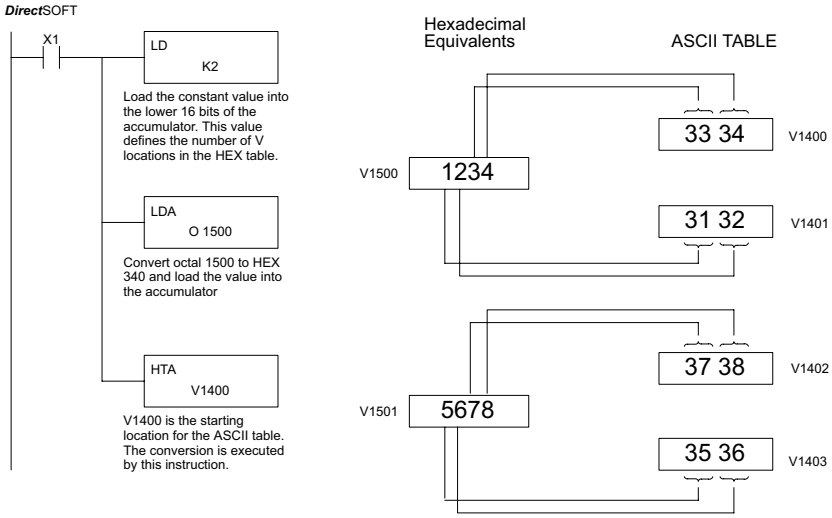
Step 2: Load the starting V-memory location for the HEX table into the accumulator. This parameter must be a HEX value.

Step 3: Specify the starting V-memory location (Vaaa) for the ASCII table in the HTA instruction.

Helpful hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
		aaa	aaa
V-memory	V	All (See page 3-56)	All (See page 3-57)

In the following example, when X1 is ON, the constant (K2) is loaded into the accumulator using the Load instruction. The starting location for the HEX table (V1500) is loaded into the accumulator using the Load Address instruction. The starting location for the ASCII table (V1400) is specified in the HEX to ASCII instruction.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT							
SHFT	L ANDST	D 3	→	PREV	C 2	ENT				
SHFT	L ANDST	D 3	A 0	→	B 1	F 5	A 0	A 0	ENT	
SHFT	H 7	T MLR	A 0	→	B 1	E 4	A 0	A 0	ENT	

The table below lists valid ASCII values for HTA conversion.

ASCII Values Valid for HTA Conversion			
Hex Value	**ASCII Value**	**Hex Value**	**ASCII Value**
0	30	8	38
1	31	9	39
2	32	A	41
3	33	B	42
4	34	C	43
5	35	D	44
6	36	E	45
7	37	F	46

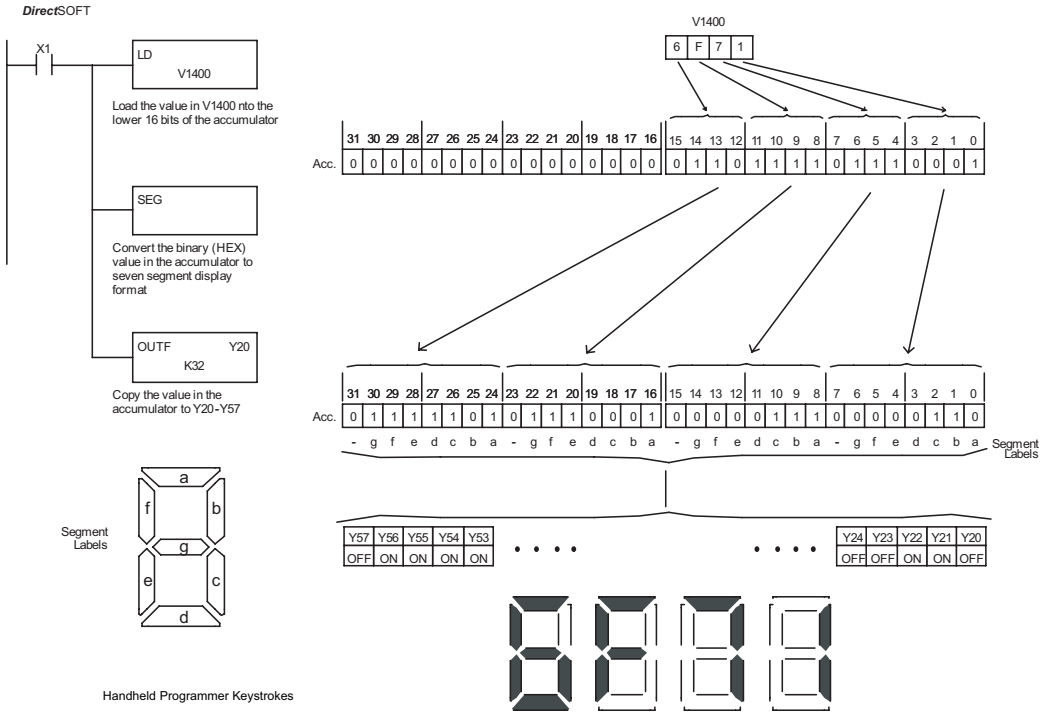
Segment (SEG)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The BCD / Segment instruction converts a 4digit HEX value in the accumulator to a 7-segment display format. The result resides in the accumulator.

In the following example, when X1 is on, the value in V1400 is loaded into the lower 16 bits of the accumulator using the Load instruction. The binary (HEX) value in the accumulator is converted to 7-segment format using the Segment instruction. The bit pattern in the accumulator is copied to Y20–Y57 using the Out Formatted instruction.



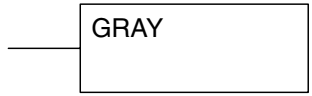
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
L ANDST	D 3	→	B 1	E 4	A 0	A 0	ENT	
SHFT	S RST	SHFT	E 4	G 6	ENT			
GX OUT	SHFT	F 5	→	C 2	A 0	→	D 3	C 2	ENT

Gray Code (GRAY)

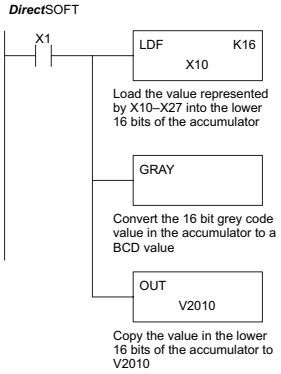
☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

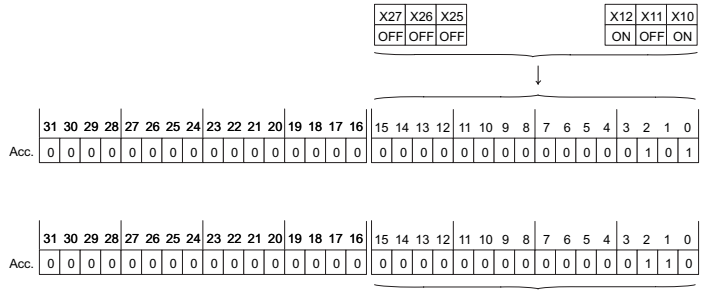
DS	Used
HPP	Used



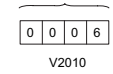
The Gray code instruction converts a 16-bit gray code value to a BCD value. The BCD conversion requires 10 bits of the accumulator. The upper 22 bits are set to "0." This instruction is designed for use with devices (typically encoders) that use the gray code numbering scheme. The Gray Code instruction will directly convert a gray code number to a BCD number for devices having a resolution of 512 or 1024 counts per revolution. If a device having a resolution of 360 counts per revolution is to be used, you must subtract a BCD value of 76 from the converted value to obtain the proper result. For a device having a resolution of 720 counts per revolution, you must subtract a BCD value of 152.

In the following example, when X1 is ON the binary value represented by X10–X27 is loaded into the accumulator using the Load Formatted instruction. The gray code value in the accumulator is converted to BCD using the Gray Code instruction. The value in the lower 16 bits of the accumulator is copied to V2010.





Gray Code	BCD
0000000000	0000
0000000001	0001
0000000011	0002
0000000010	0003
0000000110	0004
0000000111	0005
0000000101	0006
0000000100	0007
.	.
.	.
.	.
1000000001	1022
1000000000	1023



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT											
SHFT	L ANDST	D 3	F 5	→	B 1	A 0	→	B 1	G 6	ENT				
SHFT	G 6	R ORN	A 0	Y MLS	ENT									
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT						

Shuffle Digits (SFLDGT)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Shuffle Digits instruction shuffles a maximum of 8 digits rearranging them in a specified order. This function requires parameters to be loaded into the first level of the accumulator stack and the accumulator with two additional instructions. Listed below are the steps necessary to use the Shuffle Digit function. The example on the following page shows a program for the Shuffle Digits function.

SFLDGT

DS	Used
HPP	Used

Step 1: Load the value (digits) to be shuffled into the first level of the accumulator stack.

Step 2: Load the order that the digits will be shuffled to into the accumulator.

Step 3: Insert the SFLDGT instruction.

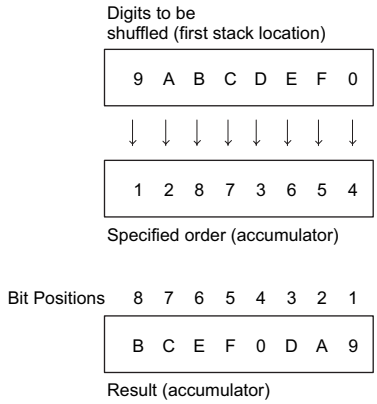
NOTE: *If the number used to specify the order contains a 0 or 9–F, the corresponding position will be set to 0.*

See example on the next page.

NOTE: *If the number used to specify the order contains duplicate numbers, the most significant duplicate number is valid. The result resides in the accumulator.*

Shuffle Digits Block Diagram

A maximum of 8 digits can be shuffled. The bit positions in the first level of the accumulator stack define the digits to be shuffled. They correspond to the bit positions in the accumulator that define the order in which the digits will be shuffled. The digits are shuffled and the result resides in the accumulator.

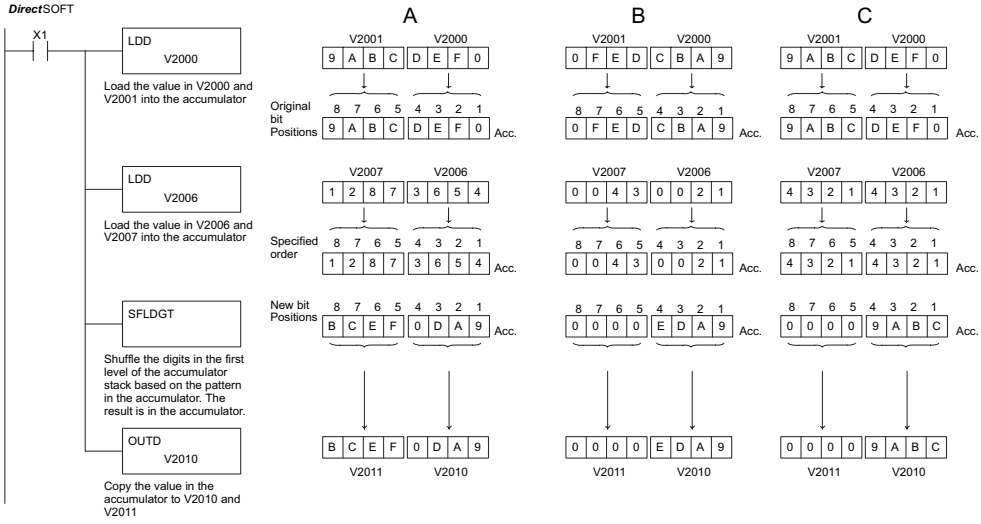


In the following examples, when input X1 is on, the value in the first level of the accumulator stack will be reorganized in the order specified by the value in the accumulator.

Example A shows how the shuffle digits works when 0 or 9 –F is not used when specifying the order the digits are to be shuffled. Also, there are no duplicate numbers in the specified order.

Example B shows how the shuffle digits works when a 0 or 9–F is used when specifying the order the digits are to be shuffled. Notice when the Shuffle Digits instruction is executed, the bit positions in the first stack location that had a corresponding 0 or 9–F in the accumulator (order specified) are set to "0".

Example C shows how the shuffle digits works when duplicate numbers are used specifying the order the digits are to be shuffled. Notice when the Shuffle Digits instruction is executed, the most significant duplicate number in the order specified is used in the result.



Handheld Programmer Keystrokes

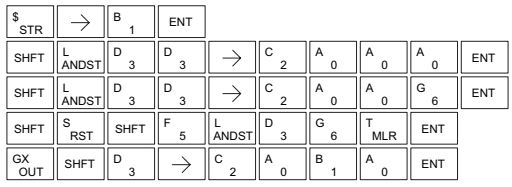


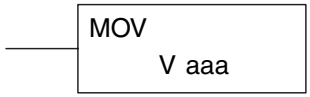
Table Instructions

Move (MOV)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Move instruction moves the values from a V-memory table to another V-memory table the same length. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Move function.



Step 1: Load the number of V-memory locations to be moved into the first level of the accumulator stack. This parameter is a HEX value (KFFF max, 7777 octal).

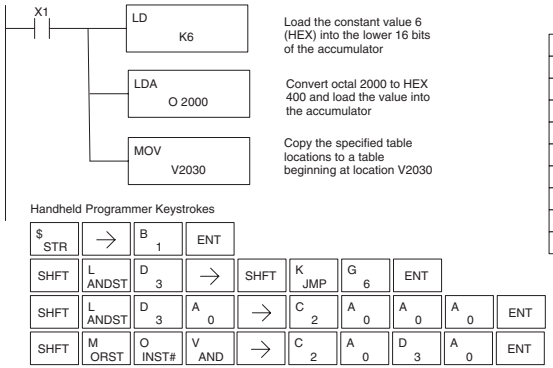
Step 2: Load the starting V-memory location for the locations to be moved into the accumulator. This parameter must be a HEX value.

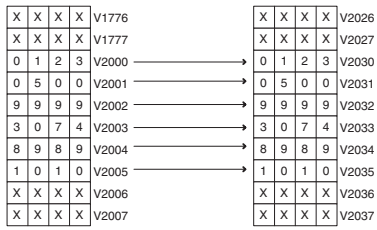
Step 3: Insert the MOVE instruction which specifies starting V-memory location (Vaaa) for the destination table.

Helpful hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
V-memory	V	All (See page 3-54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)
Pointer	P	All (See page 3-54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 2000 (V2000), the starting location for the source table is loaded into the accumulator. The destination table location (V2030) is specified in the Move instruction.





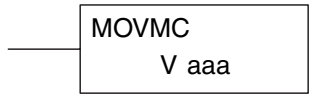
Move Memory Cartridge (MOVMC)

Load Label (LDLBL)

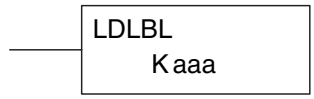
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Move Memory Cartridge instruction is used to copy data between V-memory and program ladder memory. The Load Label instruction is *only* used with the MOVMC instruction when copying data *from* program ladder memory to V-memory.



To copy data between V-memory and program ladder memory, the function parameters are loaded into the first two levels of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Move Memory Cartridge and Load Label functions.



Step 1: Load the number of words to be copied into the second level of the accumulator stack.

Step 2: Load the offset for the data label area in the program ladder memory and the beginning of the V-memory block into the first level of the accumulator stack.

Step 3: Load the *source data label* (LDLBL Kaaa) into the accumulator when copying data from ladder memory to V-memory. Load the source address into the accumulator when copying data from V-memory to ladder memory. This is where the value will be copied from. If the source address is a V-memory location, the value must be entered in HEX.

Step 4: Insert the MOVMC instruction which specifies destination (Aaaa). This is where the value will be copied to.

Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
V-memory	V	All (See page 3-54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)
Constant	K	K1-KFFFF	K1-KFFFF	K1-KFFFF	K1-KFFFF

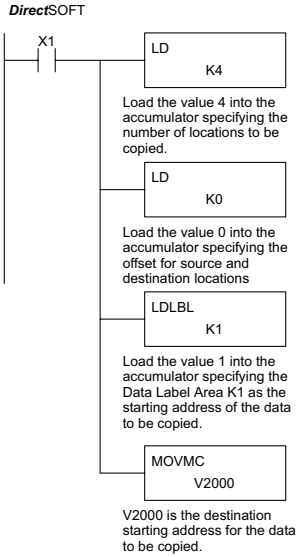
WARNING: The offset for this usage of the instruction starts at 0, but may be any number that *does not* result in data outside of the source data area being copied into the destination table. When an offset is outside of the source information boundaries, then unknown data values will be transferred into the destination table.

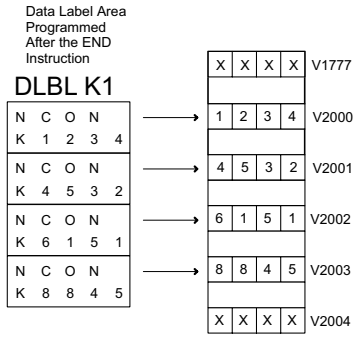
Copy Data From a Data Label Area to V-Memory

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

In the following example, data is copied from a Data Label Area to V-memory. When X1 is on, the constant value (K4) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the second stack location after the next Load and Load Label (LDLBL) instructions are executed. The constant value (K0) is loaded into the accumulator using the Load instruction. This value specifies the offset for the source and destination data, and is placed in the first stack location after the LDLBL instruction is executed. The source address where data is being copied from is loaded into the accumulator using the LDLBL instruction. The MOVMC instruction specifies the destination starting location and executes the copying of data from the Data Label Area to V-memory.





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT									
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	ENT					
SHFT	L ANDST	D 3	→	SHFT	K JMP	A 0	ENT					
SHFT	L ANDST	D 3	L ANDST	B 1	L ANDST	→	B 1	ENT				
SHFT	M ORST	O INST#	V AND	M ORST	C 2	→	C 2	A 0	A 0	A 0	ENT	

WARNING: The offset for this usage of the instruction starts at 0, but may be any number that does not result in data outside of the source data area being copied into the destination table. When an offset is outside of the source information boundaries, then unknown data values will be transferred into the destination table.

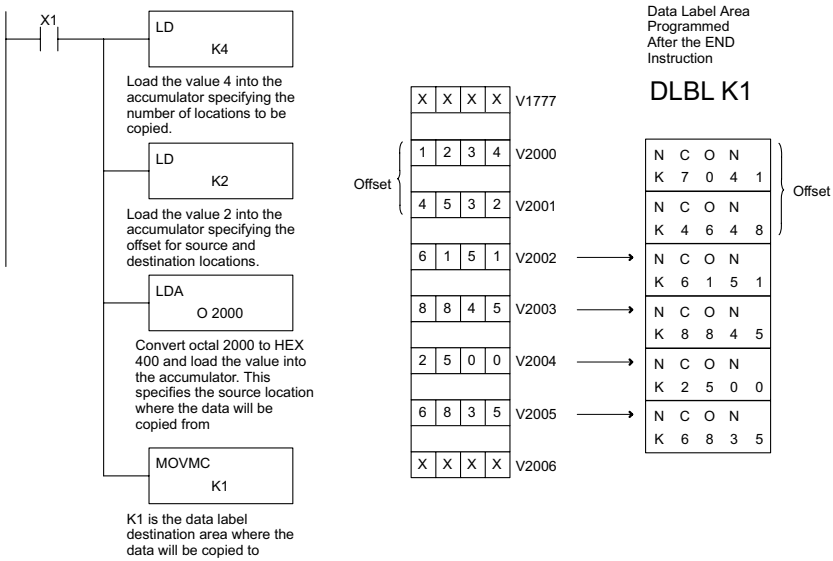
Copy Data From V-Memory to a Data Label Area

☑ 230
☑ 240
☒ 250-1
☒ 260

DS	Used
HPP	Used

In the following example, data is copied from V-memory to a data label area. When X1 is on, the constant value (K4) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the second stack location after the next Load and Load Address instructions are executed. The constant value (K2) is loaded into the accumulator using the Load instruction. This value specifies the offset for the source and destination data, and is placed in the first stack location after the Load Address instruction is executed. The source address where data is being copied from is loaded into the accumulator using the Load Address instruction. The MOVMC instruction specifies the destination starting location and executes the copying of data from V-memory to the data label area.

*Direct*SOFT



Handheld Programmer Keystrokes

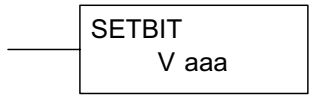
$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	ENT		
SHFT	L ANDST	D 3	→	SHFT	K JMP	C 2	ENT		
SHFT	L ANDST	D 3	A 0	→	C 2	A 0	A 0	A 0	ENT
SHFT	M ORST	O INST#	V AND	M ORST	C 2	→	SHFT	K JMP	B 1	ENT

WARNING: The offset for this usage of the instruction starts at 0. If the offset (or the specified data table range) is large enough to cause data to be copied from V-memory to beyond the end of the DLBL area, then anything after the specified DLBL area will be replaced with invalid instructions.

Set Bit (SETBIT)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

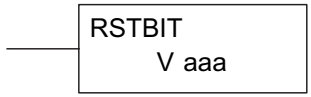
The Set Bit instruction sets a single bit to one within a range of V-memory locations.



Reset Bit (RSTBIT)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

The Reset Bit instruction resets a single bit to zero within a range of V-memory locations.



DS	Used
HPP	Used

The following description applies to both the Set Bit and Reset Bit table instructions.

Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

Step 3: Insert the Set Bit or Reset Bit instruction. This specifies the reference for the bit number of the bit you want to set or reset. The bit number is in octal, and the first bit in the table is number "0."

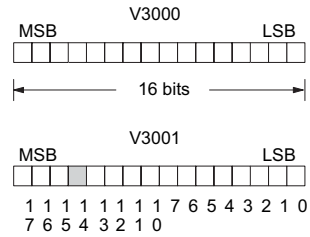
Helpful hint: — Remember that each V-memory location contains 16 bits. So, the bits of the first word of the table are numbered from 0 to 17 octal. For example, if the table length is 6 words, then 6 words = (6 x 16) bits, = 96 bits (decimal), or 140 octal. The permissible range of bit reference numbers would be 0 to 137 octal. Flag 53 will be set if the bit specified is outside the range of the table.

Operand Data Type		Range D2-260/D2-262
		aaa
V-memory	V	All (See page 3-57)

Discrete Bit Flags	Description
SP53	On when the bit number which is referred in the Set Bit or Reset Bit exceeds the range of the table

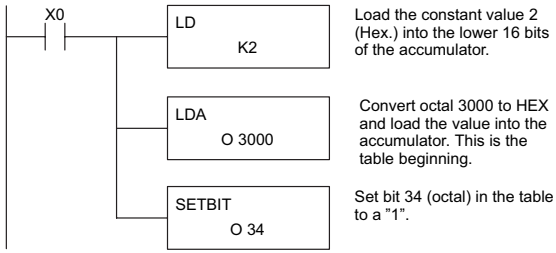
NOTE: *Status flags are only valid until the end of the scan or another instruction that uses the same flag is executed.*

For example, suppose we have a table starting at V3000 that is two words long, as shown to the right. Each word in the table contains 16 bits, or 0 to 17 in octal. To set bit 12 in the second word, we use its octal reference (bit 14). Then we compute the bit's octal address from the start of the table, so 17 + 14 = 34 octal. The following program shows how to set the bit as shown to a "1."



In this ladder example, we will use input X0 to trigger the Set Bit operation. First, we will load the table length (two words) into the accumulator stack. Next, we load the starting address into the accumulator. Since V3000 is an octal number we have to convert it to hex by using the LDA command. Finally, we use the Set Bit (or Reset Bit) instruction and specify the octal address of the bit (bit 34), referenced from the table beginning.

DirectSOFT



Handheld Programmer Keystrokes

$ STR	→	A 0	ENT						
SHFT	L ANDST	D 3	→	PREV	C 2	ENT			
SHFT	L ANDST	D 3	A 0	→	D 3	A 0	A 0	A 0	ENT
X SET	SHFT	B 1	I 8	T MLR	NEXT	D 3	E 4	ENT	

Fill (FILL)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

FILL
A aaa

The Fill instruction fills a table of up to 255 V-memory locations with a value (Aaaa), which is either a V-memory location or a 4-digit constant. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Fill function.

Step 1: Load the number of V-memory locations to be filled into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FFFF.

Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value.

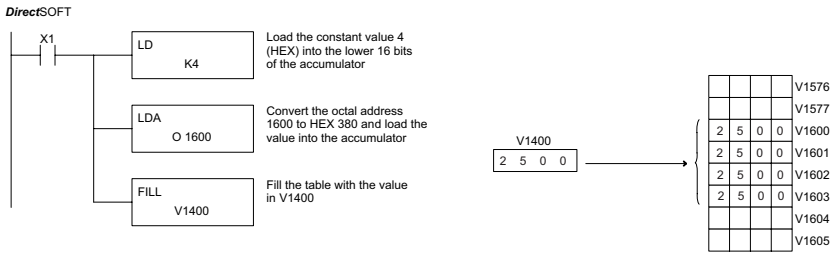
Step 3: Insert the Fill instructions which specifies the value to fill the table with.

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		Range D2-260/D2-262
	A	aaa
V-memory	V	All (See page 3-57)
Pointer	P	All V mem (See page 3-57)
Constant	K	0-FFFF

Discrete Bit Flag	Description
SP53	On if V-memory address is out of range

In the following example, when X1 is on, the constant value (K4) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed on the first level of the accumulator stack when the Load Address instruction is executed. The octal address 1600 (V1600) is the starting location for the table and is loaded into the accumulator using the Load Address instruction. The value to fill the table with (V1400) is specified in the Fill instruction.



Handheld Programmer Keystrokes

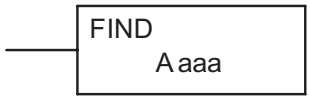
$ STR	→	B 1	ENT							
SHFT	L ANDST	D 3	→	PREV	E 4	ENT				
SHFT	L ANDST	D 3	A 0	→	B 1	G 6	A 0	A 0	ENT	
SHFT	F 5	I 8	L ANDST	L ANDST	→	B 1	E 4	A 0	A 0	ENT

Find (FIND)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Find instruction is used to search for a specified value in a V-memory table of up to 255 locations. The function parameters are loaded into the first and second levels of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Find function.



Step 1: Load the length of the table (number of V-memory locations) into the second level of the accumulator stack. This parameter must be a HEX value, 0 to FFFF.

Step 2: Load the starting V-memory location for the table into the first level of the accumulator stack. This parameter must be a HEX value.

Step 3: Load the offset from the starting location to begin the search. This parameter must be a HEX value.

Step 4: Insert the Find instruction which specifies the first value to be found in the table.

Results: The offset from the starting address to the first V-memory location which contains the search value is returned to the accumulator as a HEX value. SP53 will be set on if an address outside the table is specified in the offset or the value is not found. If the value is not found, 0 will be returned in the accumulator.

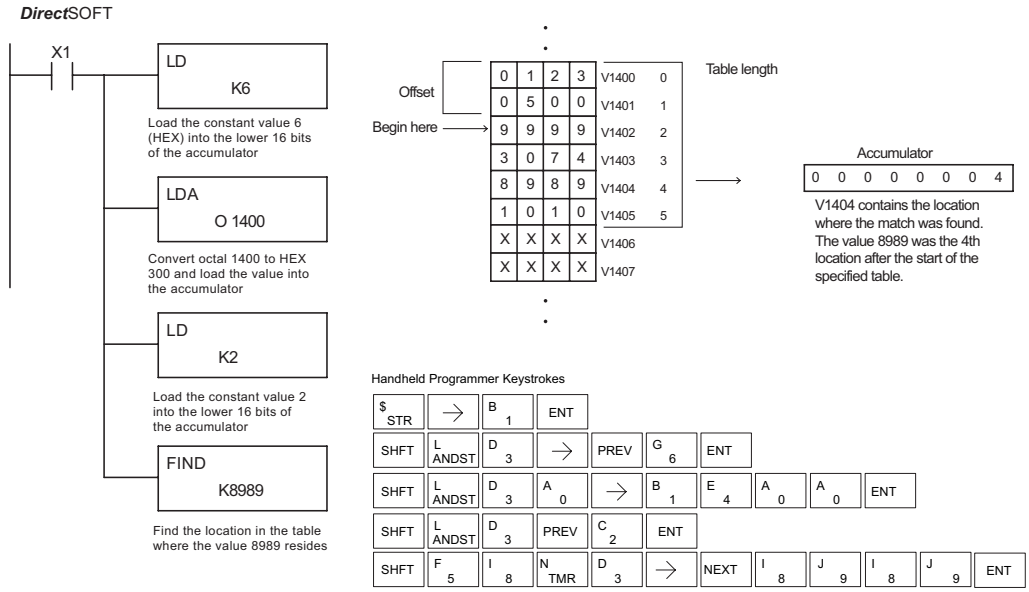
Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		Range D2-260/D2-262
A		aaa
V-memory	V	All (See page 3-57)
Constant	K	0-FFFF

Discrete Bit Flag	Description
SP53	On if there is no value in the table that is equal to the search value.

NOTE: *Status flags are only valid until another instruction that uses the same flags is executed. The pointer for this instruction starts at 0 and resides in the accumulator.*

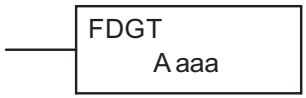
In the example on the following page, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the second stack location when the following Load Address and Load instruction is executed. The octal address 1400 (V1400) is the starting location for the table and is loaded into the accumulator. This value is placed in the first level of the accumulator stack when the following Load instruction is executed. The offset (K2) is loaded into the lower 16 bits of the accumulator using the Load instruction. The value to be found in the table is specified in the Find instruction. If a value is found equal to the search value, the offset (from the starting location of the table) where the value is located will reside in the accumulator.



Find Greater Than (FDGT)

230
240
250-1
260
262

The Find Greater Than instruction is used to search for the first occurrence of a value in a V-memory table that is greater than the specified value (Aaaa), which can be either a V-memory location or a 4-digit constant. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Find Greater Than function.



NOTE: *This instruction does not have an offset, such as the one required for the FIND instruction.*

DS	Used
HPP	Used

Step 1: Load the length of the table (up to 255 locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FFFF.

Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value.

Step 3: Insert the FDGT instruction which specifies the greater than search value.

Results: The offset from the starting address to the first V-memory location which contains the greater than search value is returned to the accumulator as a HEX value. SP53 will be set on if the value is not found and 0 will be returned in the accumulator.

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

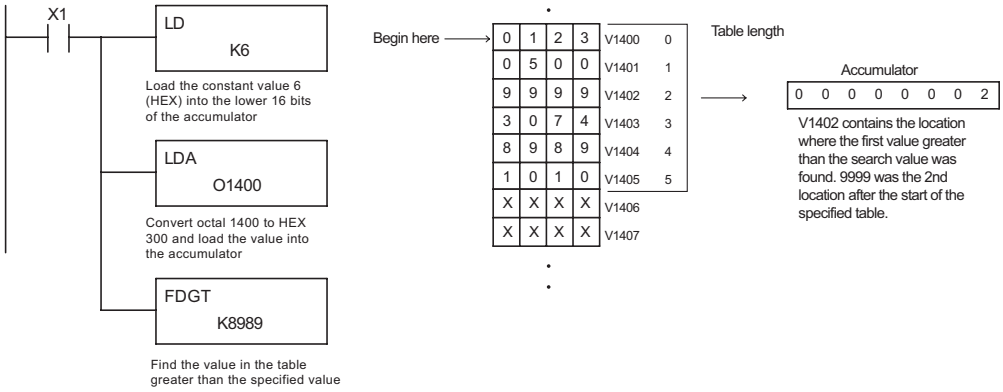
Operand Data Type		Range D2-260/D2-262
	A	aaa
V-memory	V	All (See page 3-57)
Constant	K	0-FFFF

Discrete Bit Flags	Description
SP53	On if there is no value in the table that is equal to the search value.

NOTE: *Status flags are only valid until another instruction that uses the same flags is executed. The pointer for this instruction starts at 0 and resides in the accumulator.*

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400) is the starting location for the table and is loaded into the accumulator. The greater than search value is specified in the Find Greater Than instruction. If a value is found greater than the search value, the offset (from the starting location of the table) where the value is located will reside in the accumulator. If there is no value in the table that is greater than the search value, a zero is stored in the accumulator and SP53 will come ON.

DirectSOFT



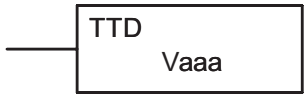
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT							
SHFT	L ANDST	D 3	→	PREV	G 6	ENT				
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT	
SHFT	F 5	D 3	G 6	T MLR	→	NEXT	I 8	J 9	I 8	J 9	ENT

Table to Destination (TTD)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used



The Table To Destination instruction moves a value from a V-memory table to a V-memory location and increments the table pointer by 1. The first V-memory location in the table contains the table pointer which indicates the next location in the table to be moved. The instruction will be executed once per scan provided the input remains on. The table pointer will reset to 1 when the value equals the last location in the table. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Table To Destination function.

Step 1: Load the length of the data table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. (Remember, the starting location of the table is used as the table pointer.) This parameter must be a HEX value.

Step 3: Insert the TTD instruction that specifies the destination V-memory location (Vaaa).

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful hint: The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful hint: The pointer location should be set to the value where the table operation will begin. The special relay SP0 or a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

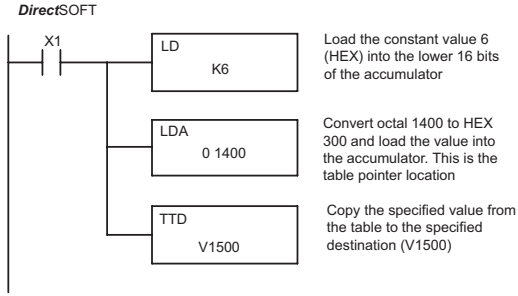
Operand Data Type		Range D2-260/D2-262
		aaa
V-memory	V	All (See page 3 - 56)

Discrete Bit Flags	Description
SP53	On if there is no value in the table that is equal to the search value.

NOTE: *Status flags (SPs) are only valid until:*
— another instruction that uses the same flag is executed, or
— the end of the scan.

The pointer for this instruction starts at 0 and resets when the table length is reached. At first glance it may appear that the pointer should reset to 0. However, it resets to 1, not 0.

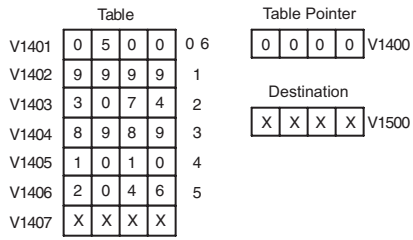
In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400) is the starting location for the source table and is loaded into the accumulator. Remember, V1400 is used as the pointer location, and is not actually part of the table data source. The destination location (V1500) is specified in the Table to Destination instruction. The table pointer (V1400 in this case) will be increased by "1" after each execution of the TTD instruction.



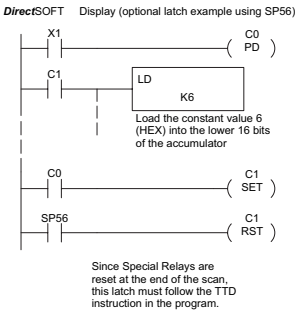
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	PREV	G 6	ENT			
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT
SHFT	T MLR	T MLR	D 3	→	B 1	F 5	A 0	A 0	ENT

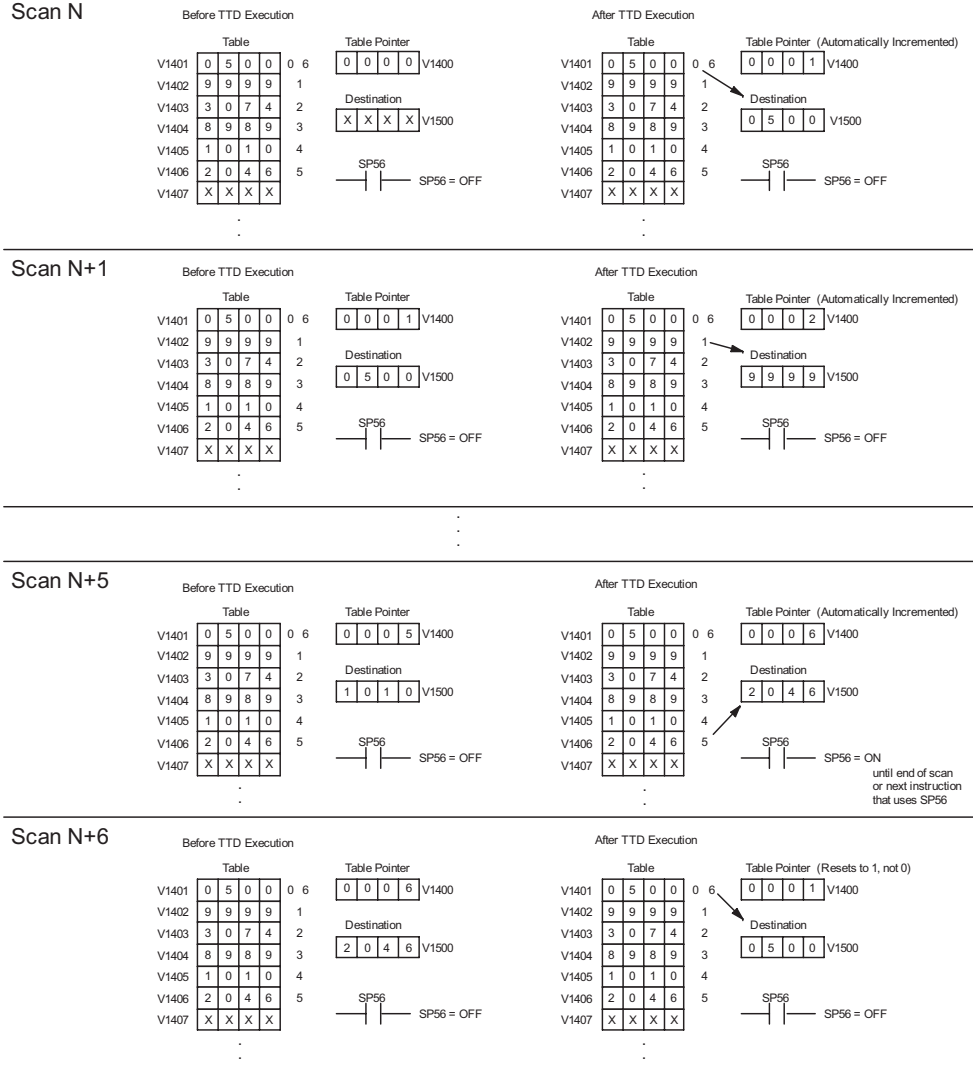
It is important to understand how the table locations are numbered. If you examine the example table, you'll notice that the first data location, V1401, will be used when the pointer is equal to 0, and again when the pointer is equal to 6. Why? Because the pointer is only equal to 0 before the very first execution. From then on, it increments from 1 to 6, and then resets to 1.



Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer increments automatically, the table would cycle through the locations very quickly. If this is a problem, you have an option of using SP56 in conjunction with a one-shot (PD) and a latch (C1 for example) to allow the table to cycle through all locations one time and then stop. The logic shown here is not required, it's just an optional method.



The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically cycles from 0 to 6, and then starts over at 1 instead of 0. Also, notice how SP56 is only on until the end of the scan.



Remove from Bottom (RFB)

☒ 230
☒ 240
☒ 250-1
☑ **260**
☑ **262**

DS	Used
HPP	Used

The Remove From Bottom instruction moves a value from the bottom of a V-memory table to a V-memory location and decrements a table pointer by 1. The first V-memory location in the table contains the table pointer which indicates the next location in the table to be moved. The instruction will be executed once per scan provided the input remains on. The instruction will stop operation when the pointer equals 0. The function parameters are loaded into the first level of the accumulator stack and the accumulator by 2 additional instructions. Listed below are the steps necessary to program the Remove From Bottom function.

Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. (Remember, the starting location of the table blank is used as the table pointer.) This parameter must be a HEX value.

Step 3: Insert the RFB instructions which specifies destination V-memory location (Vaaa).

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful hint: The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful hint: The pointer location should be set to the value where the table operation will begin. The special relay SP0 or a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

Operand Data Type		Range D2-260/D2-262
		aaa
V-memory	V	All (See page 3-57)

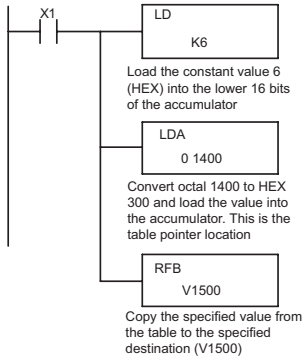
Discrete Bit Flags	Description
SP56	On when the table pointer equals 0

NOTE: *Status flags (SPs) are only valid until:*
— another instruction that uses the same flag is executed, or
— the end of the scan.

The pointer for this instruction can be set to start anywhere in the table. It is not set automatically. You have to load a value into the pointer somewhere in your program.

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400) is the starting location for the source table and is loaded into the accumulator. Remember, V1400 is used as the pointer location, and is not actually part of the table data source. The destination location (V1500) is specified in the Remove From Bottom. The table pointer (V1400 in this case) will be decremented by "1" after each execution of the RFB instruction.

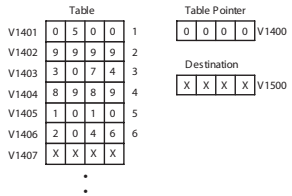
DirectSOFT



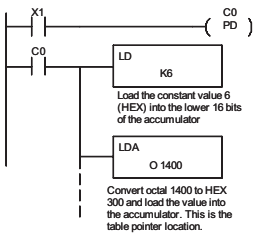
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	PREV	G 6	ENT			
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT
SHFT	R ORN	F 5	B 1	→	B 1	F 5	A 0	A 0	ENT

remove one value each time the input contact transitions from low to high.



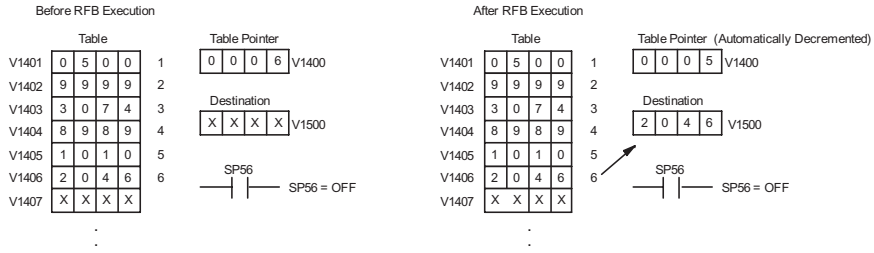
DirectSOFT (optional one-shot method)



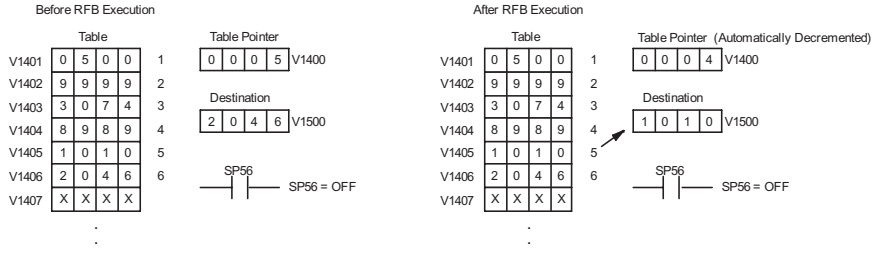
The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically decrements from 6 to 0. Also, notice how SP56 is only on until the end of the scan.

Example of Execution

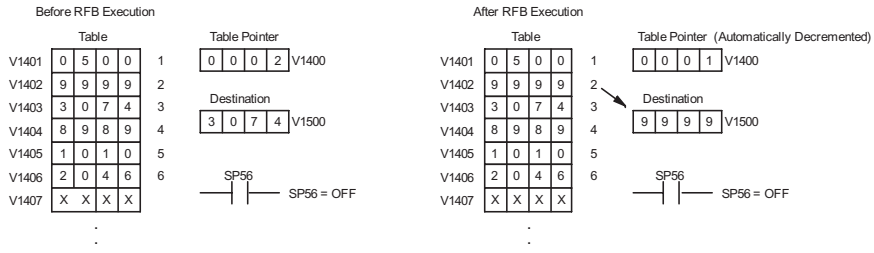
Scan N



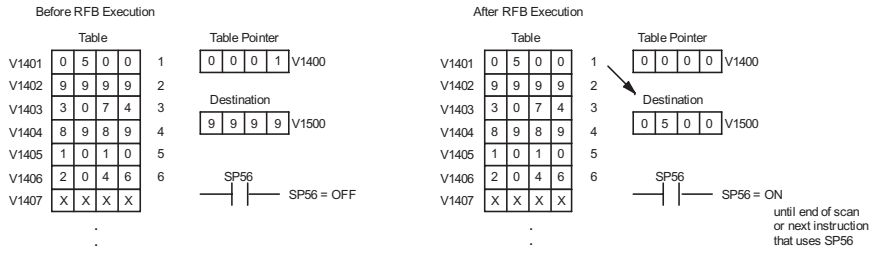
Scan N+1



Scan N+4



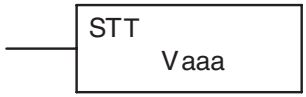
Scan N+5



Source to Table (STT)

☒ 230
☒ 240
☒ 250-1
☑ **260**
☑ **262**

DS	Used
HPP	Used



The Source To Table instruction moves a value from a V-memory location into a V-memory table and increments a table pointer by 1. When the table pointer reaches the end of the table, it resets to 1. The first V-memory location in the table contains the table pointer which indicates the next location in the table to store a value. The instruction will be executed once per scan provided the input remains on. The function parameters are loaded into the first level of the accumulator stack and the accumulator with two additional instructions. Listed below are the steps necessary to program the Source To Table function.

Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2:Load the starting V-memory location for the table into the accumulator. (Remember, the starting location of the table is used as the table pointer.) This parameter must be a HEX value.

Step 3:Insert the STT instruction which specifies the source V-memory location (Vaaa). This is where the value will be moved from.

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful hint: The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful hint: The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the pointer should be between 0 and 6. If the value is outside of this range, the data will not be moved. Also, a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

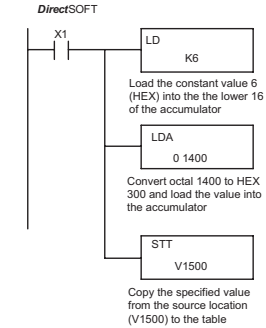
Operand Data Type		Range D2-260/D2-262
		aaa
V-memory	V	All (See page 3-57)

Discrete Bit Flags	Description
SP56	On when the table pointer equals the table length.

NOTE: *Status flags (SPs) are only valid until:*
— another instruction that uses the same flag is executed, or
— the end of the scan

The pointer for this instruction starts at 0 and resets to 1 automatically when the table length is reached.

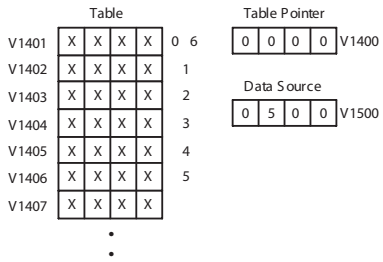
In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400), which is the starting location for the destination table and table pointer, is loaded into the accumulator. The data source location (V1500) is specified in the Source to Table instruction. The table pointer will be increased by "1" after each time the instruction is executed.



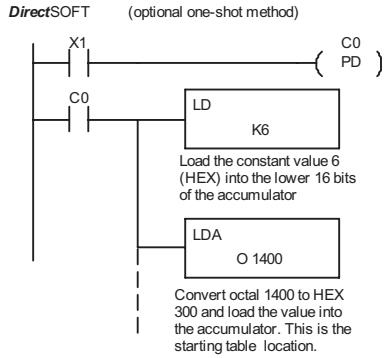
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	PREV	G 6	ENT			
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT
SHFT	S RST	SHFT	T MLR	T MLR	→	B 1	F 5	A 0	A 0	ENT

It is important to understand how the table locations are numbered. If you examine the example table, you'll notice that the first data storage location, V1401, will be used when the pointer is equal to 0, and again when the pointer is equal to 6. Why? Because the pointer is only equal to 0 before the very first execution. From then on, it increments from 1 to 6, and then resets to 1.

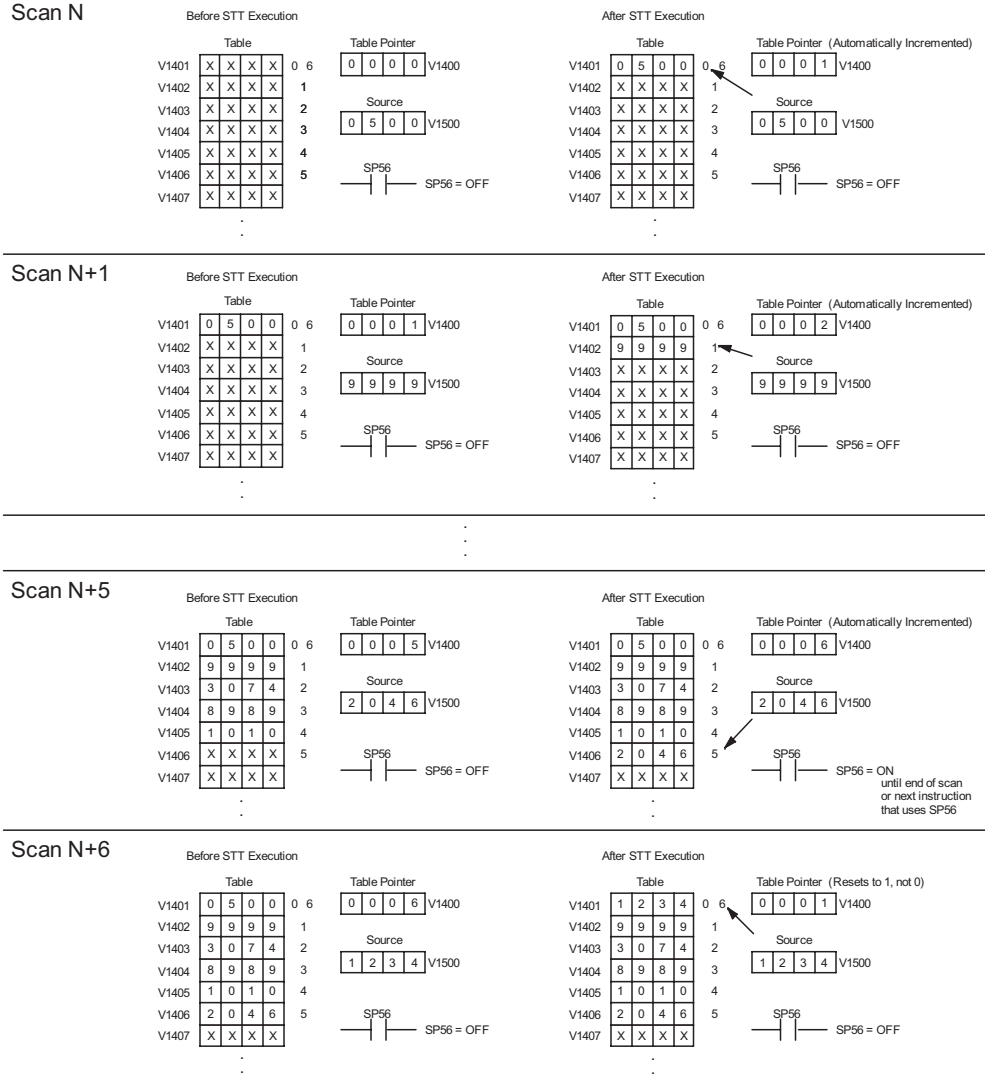


Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer increments automatically, the source data would be moved into all the table locations very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to move 1 value each time the input contact transitions from low to high.



The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically cycles from 0 to 6, and then starts over at 1 instead of 0. Also, notice how SP56 is affected by the execution. Although our example does not show it, we are assuming that there is another part of the program that changes the value in V1500 (data source) prior to the execution of the STT instruction. This is not required, but it makes it easier to see how the data source is copied into the table.

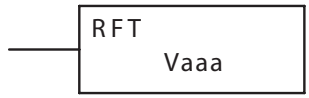
Example of Execution



Remove from Table (RFT)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used



The Remove From Table instruction pops a value off of a table and stores it in a V-memory location. When a value is removed from the table all other values are shifted up 1 location. The first V-memory location in the table contains the table length counter. The table counter decrements by 1 each time the instruction is executed. If the length counter is 0 or greater than the maximum table length (specified in the first level of the accumulator stack), the instruction will not execute and SP56 will be on.

The instruction will be executed once per scan provided the input remains on. The function parameters are loaded into the first level of the accumulator stack and the accumulator by 2 additional instructions. Listed below are the steps necessary to program the Remove From Table function.

Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. (Remember, the starting location of the table is used as the table length counter.) This parameter must be a HEX value.

Step 3: Insert the RFT instructions which specifies destination V-memory location (Vaaa). This is where the value will be moved to.

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful hint: The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

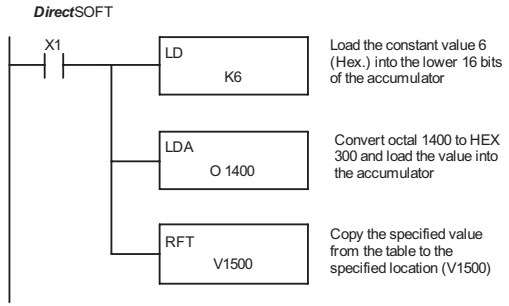
Helpful hint: The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the table counter should be between 1 and 6. If the value is outside of this range or 0, the data will not be moved from the table. Also, a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

Operand Data Type	Range D2-260/D2-262
	aaa
V-memory V	All (See page 3-57)

Discrete Bit Flags	Description
SP56	On when the table counter equals 0.

NOTE: *Status flags (SPs) are only valid until:*
— another instruction that uses the same flag is executed, or
— the end of the scan
The pointer for this instruction can be set to start anywhere in the table. It is not set automatically. You have to load a value into the pointer somewhere in your program.

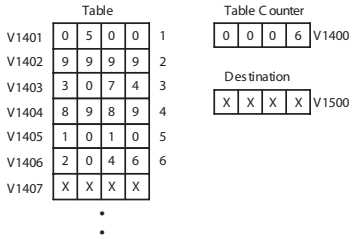
In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400) is the starting location for the source table and is loaded into the accumulator. The destination location (V1500) is specified in the Remove from Table instruction. The table counter will be decreased by "1" after the instruction is executed.



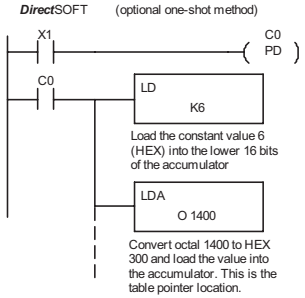
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	PREV	G 6	ENT			
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT
SHFT	R ORN	F 5	T MLR	→	B 1	F 5	A 0	A 0	ENT

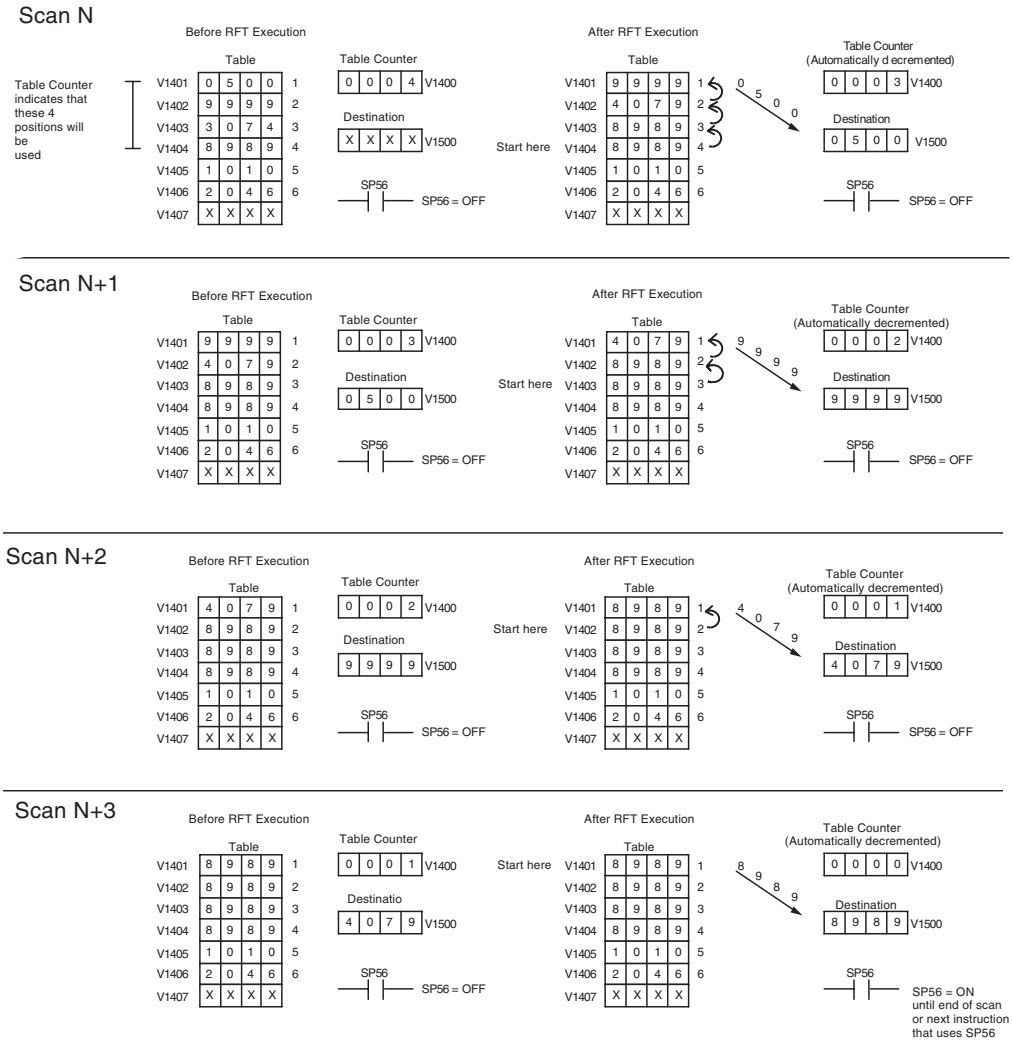
Since the table counter specifies the range of data that will be removed from the table, it is important to understand how the table locations are numbered. If you examine the example table, you'll notice that the data locations are numbered from the top of the table. For example, if the table counter started at 6, then all 6 of the locations would be affected during the instruction execution.



Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer decrements automatically, the data would be removed from the table very quickly. If this is a problem for your application, you have the option of using a one-shot (PD) to remove one value each time the input contact transitions from low to high.

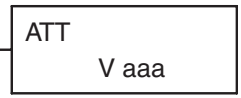


The following diagram shows the scan-by-scan results of the execution for our example program. In our example we're showing the table counter set to 4 initially (Remember, you can set the table counter to any value that is within the range of the table). The table counter automatically decrements from 4 to 0 as the instruction is executed. Notice how the last 2 table positions, 5 and 6, are not moved up through the table. Also, notice how SP56, which comes on when the table counter is 0, is only on until the end of the scan.



Add to Top (ATT)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262



The Add To Top instruction pushes a value onto a V-memory table from a V-memory location. When the value is added to the table, all other values are pushed down 1 location.

The instruction will be executed once per scan provided the input remains on. The function parameters are loaded into the first level of the accumulator stack and the accumulator by 2 additional instructions. Listed below are the steps necessary to program the Add To Top function.

DS	Used
HPP	Used

Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. (Remember, the starting location of the table is used as the table length counter.) This parameter must be a HEX value.

Step 3: Insert the ATT instruction that specifies the source V-memory location (Vaaa). This is where the value will be moved from.

Helpful hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful hint: The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful hint: The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the table counter should be between 1 and 6. If the value is outside of this range or zero, the data will not be moved into the table. Also, a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

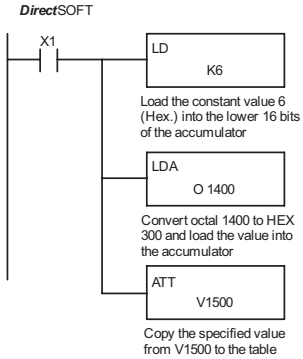
Operand Data Type		Range D2-260/D2-262
		aaa
V-memory	V	All (See page 3-57)

Discrete Bit Flags	Description
SP56	On when the table counter equals 0.

NOTE: *Status flags (SPs) are only valid until:*
— another instruction that uses the same flag is executed, or
— the end of the scan

The pointer for this instruction can be set to start anywhere in the table. It is not set automatically. You have to load a value into the pointer somewhere in your program.

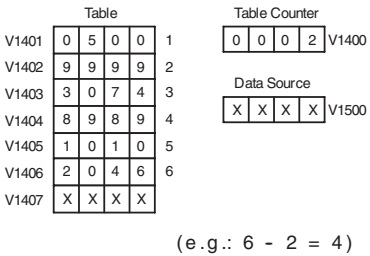
In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400), which is the starting location for the destination table and table counter, is loaded into the accumulator. The source location (V1500) is specified in the Add to Top instruction. The table counter will be increased by "1" after the instruction is executed.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	PREV	G 6	ENT		
SHFT	L ANDST	D 3	A 0	→	B 1	E 4	A 0	A 0	ENT
SHFT	A 0	T MLR	T MLR	→	B 1	F 5	A 0	A 0	ENT

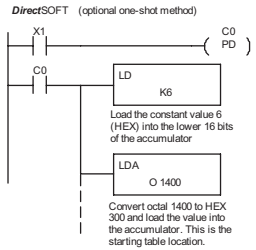
For the ATT instruction, the table counter determines the number of additions that can be made before the instruction will stop executing. So, it is helpful to understand how the system uses this counter to control the execution.



For example, if the table counter was set to 2, and the table length was 6 words, then there could only be 4 additions of data before the execution was stopped. This can be calculated easily by:

Table length – table counter = number of executions

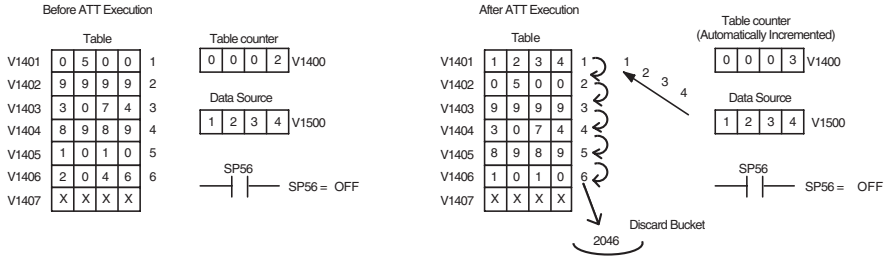
Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the table counter increments automatically, the data would be moved into the table very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to add one value each time the input contact transitions from low to high.



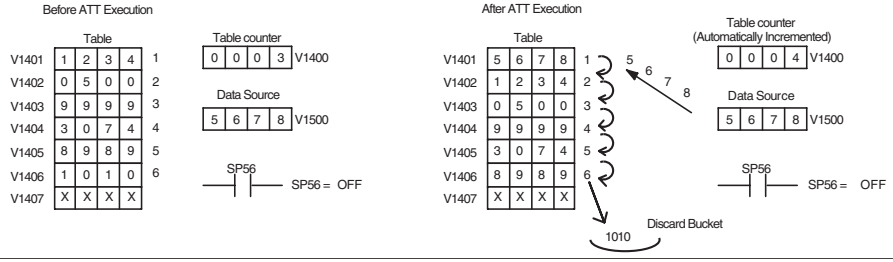
The following diagram shows the scan-by-scan results of the execution for our example program. The table counter is set to 2 initially, and it will automatically increment from 2 to 6 as the instruction is executed. Notice how SP56 comes on when the table counter is 6, which is equal to the table length. Plus, although our example does not show it, we are assuming that there is another part of the program that changes the value in V1500 (data source) prior to the execution of the ATT instruction.

Example of Execution

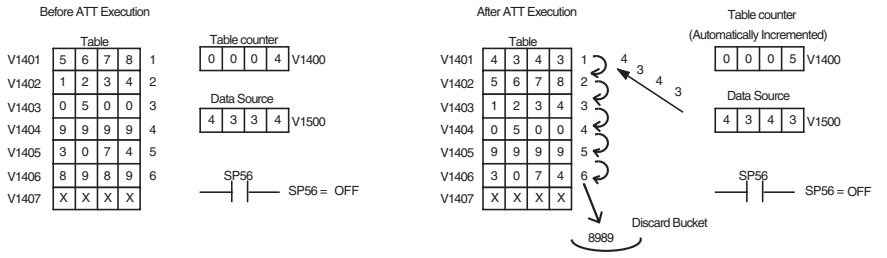
Scan N



Scan N+1



Scan N+2



Scan N+3

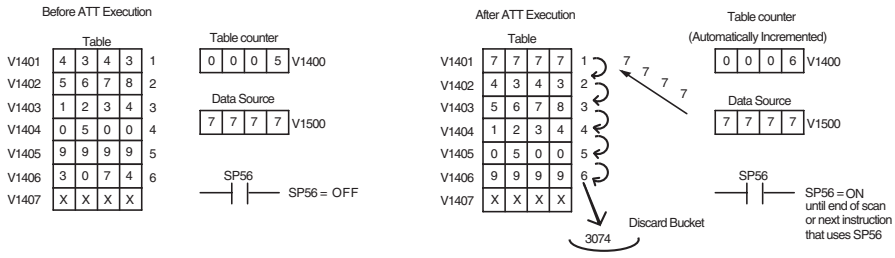


Table Shift Left (TSHFL)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

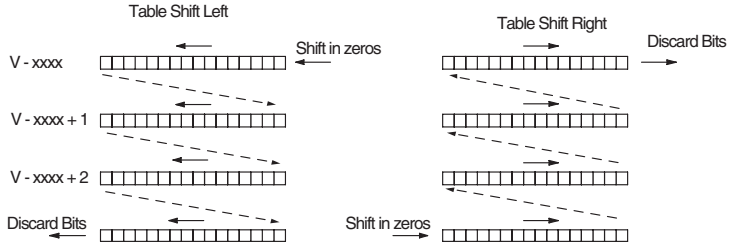
The Table Shift Left instruction shifts all the bits in a V-memory table to the left a specified number of bit positions.

Table Shift Right (TSHFR)

DS	Used
HPP	Used

The Table Shift Right instruction shifts all the bits in a V-memory table to the right a specified number of bit positions.

The following description applies to both the Table Shift Left and Table Shift Right instructions. A table is a range of V-memory locations. The Table Shift Left and Table Shift Right instructions shift bits serially throughout the entire table. Bits are shifted out the end of one word and into the opposite end of an adjacent word. At the ends of the table, bits are either discarded, or zeros are shifted into the table. The example tables below are arbitrarily four words long.



Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

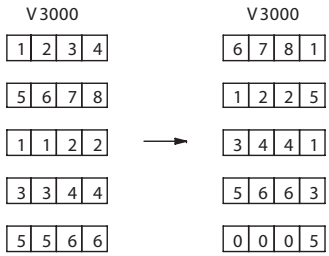
Step 3: Insert the Table Shift Left or Table Shift Right instruction. This specifies the number of bit positions you wish to shift the entire table. The number of bit positions must be in octal.

Helpful hint: Remember that each V-memory location contains 16 bits. The bits of the first word of the table are numbered from 0 to 17 octal. If you want to shift the entire table by 20 bits, that is 24 octal. Flag 53 will be set if the number of bits to be shifted is larger than the total bits contained within the table. Flag 67 will be set if the last bit shifted (just before it is discarded) is a "1".

Operand Data Type		Range D2-260/D2-262
		aaa
V-memory	V	All (See page 3-57)

Discrete Bit Flags	Description
SP53	On when the number of bits to be shifted is larger than the total bits contained within the table
SP67	On when the last bit shifted (just before it is discarded) is a "1"

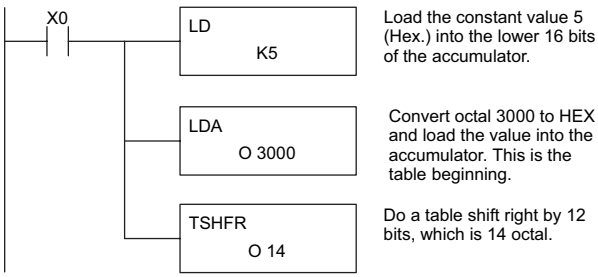
NOTE: *Status flags are only valid until:*
— the end of the scan
— or another instruction that uses the same flag is executed.



The example table to the right contains BCD data as shown (for demonstration purposes). Suppose we want to do a table shift right by 3 BCD digits (12 bits). Converting to octal, 12 bits is 14 octal. Using the Table Shift Right instruction and specifying a shift by octal 14, we have the resulting table shown at the far right. Notice that the 2–3–4 sequence has been discarded, and the 0–0–0 sequence has been shifted in at the bottom.

The following ladder example assumes the data at V3000 to V3004 already exists as shown above. We will use input X0 to trigger the Table Shift Right operation. First, we will load the table length (5 words) into the accumulator stack. Next, we load the starting address into the accumulator. Since V3000 is an octal number we have to convert it to hex by using the LDA command. Finally, we use the Table Shift Right instruction and specify the number of bits to be shifted (12 decimal), which is 14 octal.

DirectSOFT



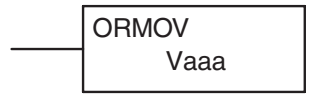
Handheld Programmer Keystrokes

$ STR	→	A 0	ENT								
SHFT	L ANDST	D 3	→	PREV	F 5	ENT					
SHFT	L ANDST	D 3	A 0	→	D 3	A 0	A 0	A 0	ENT		
SHFT	T MLR	SHFT	S RST	H 7	F 5	R ORN	→	NEXT	B 1	E 4	ENT

AND Move (ANDMOV)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

The AND Move instruction copies data from a table to the specified memory location, ANDing each word with the accumulator data as it is written.



OR Move (ORMOV)

The Or Move instruction copies data from a table to the specified memory location, ORing each word with the accumulator contents as it is written.

Exclusive OR Move (XORMOV)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

The Exclusive OR Move instruction copies data from a table to the specified memory location, XORing each word with the accumulator value as it is written.

The following description applies to the AND Move, OR Move, and Exclusive OR Move instructions. A table is just a range of V-memory locations. These instructions copy the data of a table to another specified location, preforming a logical operation on each word with the accumulator contents as the new table is written.

DS	Used
HPP	Used

Step 1: Load the length of the table (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

Step 3: Load the BCD/hex bit pattern into the accumulator which will be logically combined with the table contents as they are copied.

Step 4: Insert the AND Move, OR Move, or XOR Move instruction. This specifies the starting location of the copy of the original table. This new table will automatically be the same length as the original table.

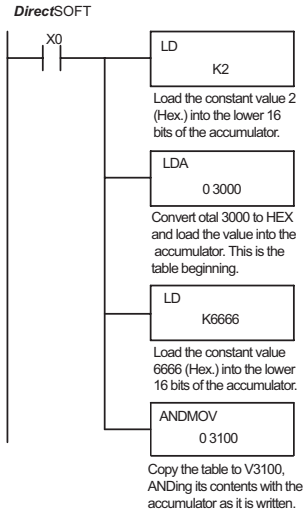
Operand Data Type		Range D2-260/D2-262
		aaa
V-memory	V	All (See page 3-57)

The example table to the right contains BCD data as shown (for demonstration purposes). Suppose we want to move a table of two words at V3000 and AND it with K6666. The copy of the table at V3100 shows the result of the AND operation for each word.

The program on the next page performs the ANDMOV operation example above. It assumes that the data in the table at V3000 – V3001 already exists. First we load the table length (two words) into the accumulator. Next we load the starting address of the source table, using the LDA instruction. Then we load the data into the accumulator to be ANDed with the table. In the ANDMOV command, we specify the table destination, V3100.

Handheld Programmer Keystrokes

$ STR	→	A 0	ENT							
SHFT	L ANDST	D 3	→	PREV	C 2	ENT				
SHFT	L ANDST	D 3	A 0	→	D 3	A 0	A 0	A 0	ENT	
SHFT	L ANDST	D 3	→	PREV	G 6	G 6	G 6	G 6	ENT	
V AND	SHFT	M ORST	O INST#	V AND	→	D 3	B 1	A 0	A 0	ENT

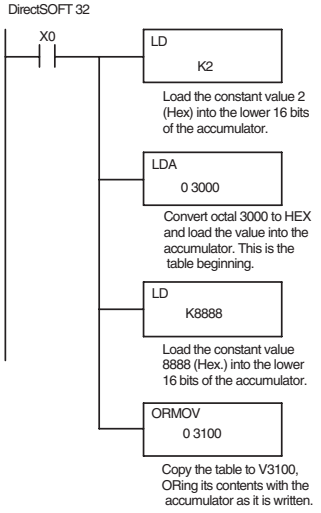


The example to the right shows a table of two words at V3000 and logically ORs it with K8888. The copy of the table at V3100 shows the result of the OR operation for each word.

The program to the right performs the ORMOV example above. It assumes that the data in the table at V3000 – V3001 already exists. First we load the table length (two words) into the accumulator. Next we load the starting address of the source table, using the LDA instruction. Then we load the data into the accumulator to be ORed with the table. In the ORMOV command, we specify the table destination, V3100.

Handheld Programmer Keystrokes

$ STR	→	A 0	ENT							
SHFT	L ANDST	D 3	→	PREV	C 2	ENT				
SHFT	L ANDST	D 3	A 0	→	D 3	A 0	A 0	A 0	ENT	
SHFT	L ANDST	D 3	→	PREV	I 8	I 8	I 8	I 8	ENT	
Q OR	SHFT	M ORST	O INST#	V AND	→	D 3	B 1	A 0	A 0	ENT



The example to the right shows a table of two words at V3000 and logical XORs it with K3333. The copy of the table at V3100 shows the result of the XOR operation for each word.

The ladder program example for the XORMOV is similar to the one above for the ORMOV. Just use the XORMOV instruction. On the Handheld Programmer, you must use the SHFT key and spell "XORMOV" explicitly.

Find Block (FINDB)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

The Find Block instruction searches for an occurrence of a specified block of values in a V-memory table. The function parameters are loaded into the first and second levels of the accumulator stack and the accumulator by three additional instructions. If the block is found, its starting address will be stored in the accumulator. If the block is not found, flag SP53 will be set.

DS	Used
HPP	N/A

Operand Data Type		Range D2-260/D2-262
	A	aaa
V-memory	V	All (See page 3 - 56)
V-memory	P	All (See page 3 - 56)

Discrete Bit Flags	Description
SP53	On when the Find Block instruction was executed but did not find the block of data in table specified.

The steps below are necessary to program the Find Block function.

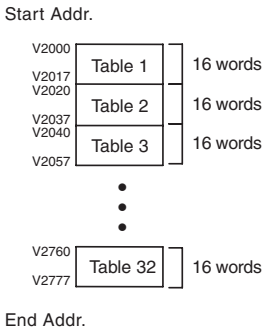
Step 1: Load the number of bytes in the block to be located. This parameter must be a decimal value from 1 to 256.

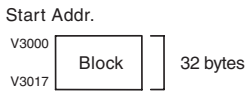
Step 2: Load the length of a table (number of words) to be searched. The Find Block will search multiple tables that are adjacent in V-memory. This parameter must be a decimal value from 1 to 128.

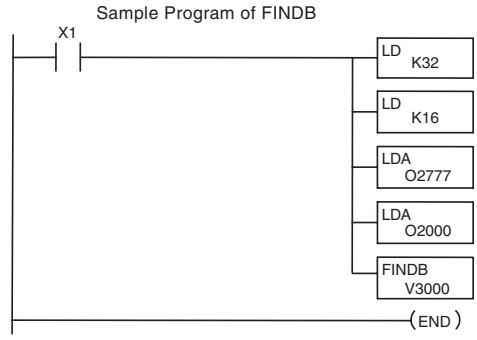
Step 3: Load the ending location for all the tables into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

Step 4: Load the table starting location for all the tables into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

Step 5: Insert the Find Block instruction. This specifies the starting location of the block of data you are trying to locate.





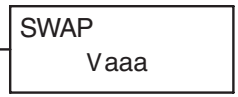


Swap (SWAP)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Swap instruction exchanges the data in two tables of equal length.



The following steps apply to both the Set Bit and Reset Bit table instructions.

Step 1: Load the length of the tables (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF. Remember that the tables must be of equal length.

Step 2: Load the starting V-memory location for the first table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

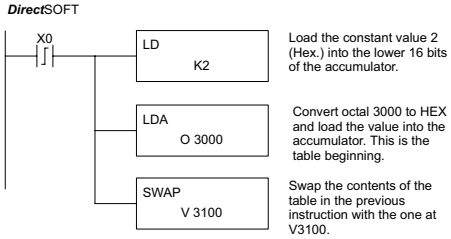
Step 3: Insert the Swap instruction. This specifies the starting address of the second table.

Helpful hint: The data swap occurs within a single scan. If the instruction executes on multiple consecutive scans, it will be difficult to know the actual contents of either table at any particular time. So, remember to swap just on a single scan.

Operand Data Type		Range D2-260/D2-262
		aaa
V-memory	V	All (See page 3-57)

The example to the right shows a table of two words at V3000. We will swap its contents with another table of two words at V3100 by using the Swap instruction.

The example program below uses a PD contact (triggers for one scan for off-to-on transition). First, we load the length of the tables (two words) into the accumulator. Then we load the address of the first table (V3000) into the accumulator using the LDA instruction, converting the octal address to hex. Note that it does not matter which table we declare "first," because the swap results will be the same.



Handheld Programmer Keystrokes

$ STR	SHFT	P CV	D 3	→	A 0	ENT			
SHFT	L ANDST	D 3	→	PREV	C 2	ENT			
SHFT	L ANDST	D 3	A 0	→	D 3	A 0	A 0	A 0	ENT
SHFT	S RST	SHFT	W ANDN	A 0	P CV	→	D 3	B 1	A 0	A 0	ENT

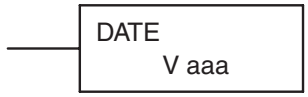
Clock/Calendar Instructions

Date (DATE)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

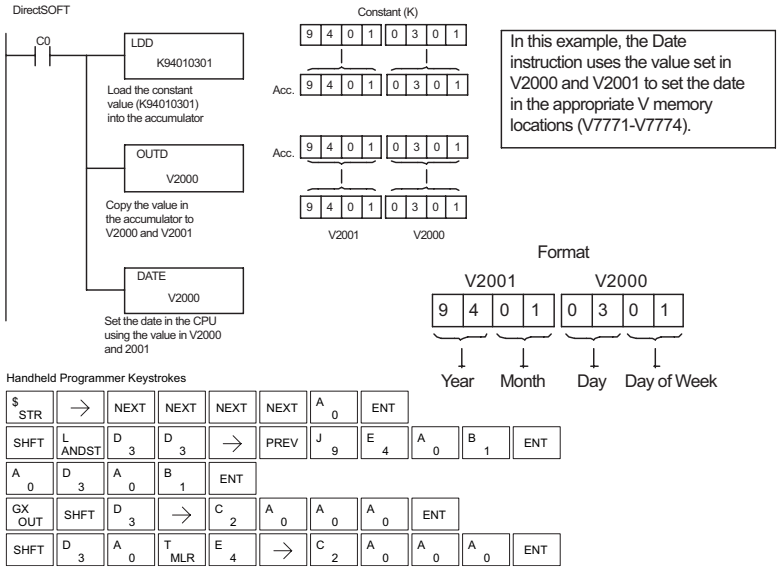
The Date instruction can be used to set the date in the CPU. The instruction requires two consecutive V-memory locations (Vaaa) to set the date. If the values in the specified locations are not valid, the date will not be set. The current date can be read from 4 consecutive V-memory locations (V7771–V7774).



Date	Range	V-memory Location (BCD) (READ Only)
Year	0-99	V7774
Month	1-12	V7773
Day	1-31	V7772
Day of Week	0-06	V7771
The values entered for the day of week are: 0=Sunday, 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday.		

Operand Data Type		Range	
		D2-250-1	D2-260/D2-262
		aaa	aaa
V-memory	V	All (See page 3-56)	All (See page 3-57)

In the following example, when C0 is on, the constant value (K94010301) is loaded into the accumulator using the Load Double instruction (C0 should be a contact from a one shot (PD) instruction). The value in the accumulator is output to V2000 using the Out Double instruction. The Date instruction uses the value in V2000 to set the date in the CPU.



Time (TIME)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

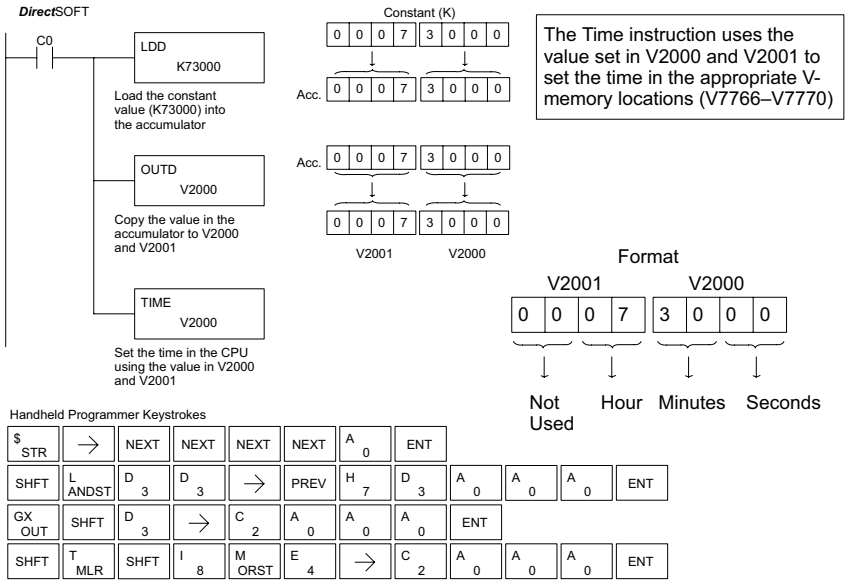
DS	Used
HPP	Used

The Time instruction can be used to set the time (24-hour clock) in the CPU. The instruction requires two consecutive V-memory locations (Vaaa) which are used to set the time. If the values in the specified locations are not valid, the time will not be set. The current time can be read from memory locations V7747 and V7766–V7770.

Date	Range	V-memory Location (BCD) (READ Only)
1/100 seconds (10ms)	0-99	V7747
Seconds	0-59	V7766
Minutes	0-59	V7767
Hour	0-23	V7770

Operand Data Type	Range	
	D2-250-1	D2-260/D2-262
	aaa	aaa
V-memory V	All (See page 3-56)	All (See page 3-57)

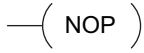
In the following example, when C0 is on, the constant value (K73000) is loaded into the accumulator using the Load Double instruction (C0 should be a contact from a one shot (PD) instruction). The value in the accumulator is output to V2000 using the Out Double instruction. The Time instruction uses the value in V2000 to set the time in the CPU.



CPU Control Instructions

No Operation (NOP)

The No Operation is an empty (not programmed) memory location.



☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

*Direct*SOFT

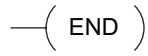
Handheld Programmer Keystrokes

SHFT	N TMR	O INST#	P CV	ENT

End (END)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The End instruction marks the termination point of the normal program scan. An End instruction is required at the end of the main program body. If the End instruction is omitted, an error will occur and the CPU will not enter the Run Mode. Data labels, subroutines and interrupt routines are placed after the End instruction. The End instruction is not conditional; therefore, no input contact is allowed.



DS	Used
HPP	Used

*Direct*SOFT

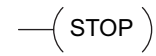
Handheld Programmer Keystrokes

SHFT	E 4	N TMR	D 3	ENT

Stop (STOP)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

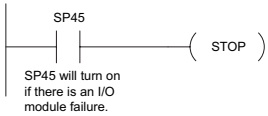
The Stop instruction changes the operational mode of the CPU from Run to Program (Stop) mode. This instruction is typically used to stop PLC operation in a shutdown condition such as an I/O module failure.



In the following example, when SP45 comes on indicating an I/O module failure, the CPU will stop operation and switch to the program mode.

DS	Used
HPP	Used

*Direct*SOFT



Handheld Programmer Keystrokes

$ STR	→	SHFT	SP STRN	E 4	F 5	ENT
SHFT	S RST	SHFT	T MLR	O INST#	P CV	ENT

Reset Watch Dog Timer (RSTWT)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Reset Watch Dog Timer instruction resets the CPU scan timer. The default setting for the watch dog timer is 200ms. Scan times very seldom exceed 200ms, but it is possible. For/next loops, subroutines, interrupt routines, and table instructions can be programmed such that the scan becomes longer than 200ms. When instructions are used in a manner that could exceed the watch dog timer setting, this instruction can be used to reset the timer.

A software timeout error (E003) will occur and the CPU will enter the program mode if the scan time exceeds the watch dog timer setting. Placement of the RSTWT instruction in the program is very important. The instruction has to be executed before the scan time exceeds the watch dog timer's setting.

If the scan time is consistently longer than the watch dog timer's setting, the timeout value may be permanently increased from the default value of 200ms by AUX 55 on the HPP or the appropriate auxiliary function in your programming package. This eliminates the need for the RSTWT instruction.

In the following example, the CPU scan timer will be reset to 0 when the RSTWT instruction is executed. See the For/Next instruction for a detailed example.

DirectSOFT

Handheld Programmer Keystrokes

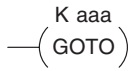
SHFT	R ORN	S RST	T MLR	W ANDN	T MLR	ENT

Program Control Instructions

Goto Label (GOTO) (LBL)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

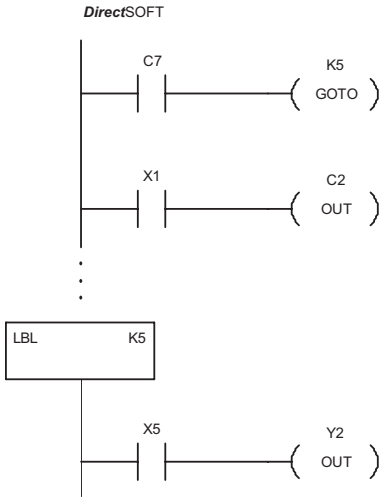
The Goto / Label skips all instructions between the Goto and the corresponding LBL instruction. The operand value for the Goto and the corresponding LBL instruction is the same. The logic between Goto and LBL instruction is not executed when the Goto instruction is enabled. Up to 128 Goto instructions and 64 LBL instructions can be used in the program.



DS	Used
HPP	Used

Operand Data Type		Range		
		D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa
Constant	K	1-FFFF	1-FFFF	1-FFFF

In the following example, when C7 is on, all the program logic between the GOTO and the corresponding LBL instruction (designated with the same constant Kaaa value) will be skipped. The instructions being skipped will not be executed by the CPU.



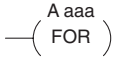
Handheld Programmer Keystrokes

$ STR	→	SHFT	C 2	H 7	ENT			
SHFT	G 6	O INST#	T MLR	O INST#	→	F 5	ENT	
$ STR	→	B 1	ENT					
GX OUT	→	SHFT	C 2	C 2	ENT			
:								
SHFT	L ANDST	B 1	L ANDST	→	F 5	ENT		
$ STR	→	F 5	ENT					
GX OUT	→	C 2	ENT					

For/Next (FOR) (NEXT)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The For and Next instructions are used to execute a section of ladder logic between the For and Next instruction a specified numbers of times. When the For instruction is enabled, the program will loop the specified number of times. If the For instruction is not energized, the section of ladder logic between the For and Next instructions is not executed.



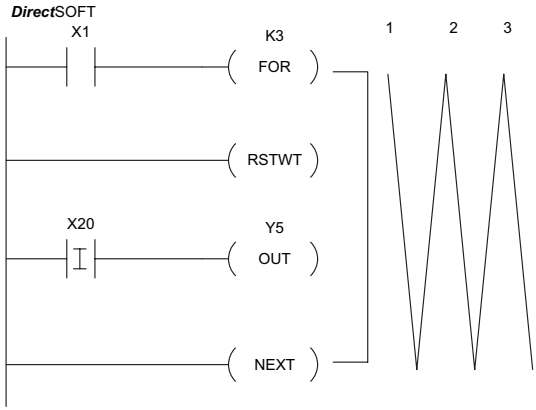
DS	Used
HPP	Used

For/Next instructions cannot be nested. Up to 64 For/Next loops may be used in a program. If the maximum number of For/Next loops is exceeded, error E413 will occur.

The normal I/O update and CPU housekeeping is suspended while executing the For/Next loop. The program scan time can increase significantly, depending on the number of times the logic between the For and Next instruction is executed. With the exception of immediate I/O instructions, I/O will not be updated until the program execution is completed for that scan. Depending on the length of time required to complete the program execution, it may be necessary to reset the watchdog timer inside of the For/Next loop using the RSTWT instruction.

Operand Data Type		Range		
		D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa
V-memory	V	All (See page 3 - 54)	All (See page 3 - 55)	All (See page 3 - 56)
Constant	K	1-9999	1-9999	1-9999

In the following example, when X1 is on, the application program inside the For/Next loop will be executed three times. If X1 is off, the program inside the loop will not be executed. The immediate instructions may or may not be necessary depending on your application. Also, The RSTWT instruction is not necessary if the For/Next loop does not extend the scan time larger the Watchdog Timer setting. For more information on the Watchdog Timer, refer to the RSTWT instruction.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT			
SHFT	F 5	O INST#	R ORN	→	D 3	ENT
SHFT	R ORN	S RST	T MLR	W ANDN	T MLR	ENT
$ STR	SHFT	I 8	→	C 2	A 0	ENT
GX OUT	→	F 5	ENT			
SHFT	N TMR	E 4	X SET	T MLR	ENT	

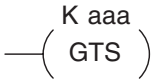
Goto Subroutine (GTS) (SBR)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Goto Subroutine instruction allows a section of ladder logic to be placed outside the main body of the program and execute only when needed. There can be a maximum of 128 GTS instructions and 64 SBR instructions used in a program. The GTS instructions can be nested up to 8 levels. An error E412 will occur if the maximum limits are exceeded. Typically this will be used in an application where a block of program logic may be slow to execute and is not required to execute every scan. The subroutine label and all associated logic is placed after the End statement in the program. When the subroutine is called from the main program, the CPU will execute the subroutine (SBR) with the same constant number (K) as the GTS instruction that called the subroutine.

By placing code in a subroutine, it is only scanned and executed when needed since it resides after the End instruction. Code which is not scanned does not impact the overall scan time of the program.



Operand Data Type		Range		
		D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa
Constant	K	1-FFFF	1-FFFF	1-FFFF

Subroutine Return (RT)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

When a Subroutine Return is executed in the subroutine, the CPU will return to the point in the main body of the program from which it was called. The Subroutine Return is used as termination of the subroutine, which must be the last instruction in the subroutine and is a stand-alone instruction (no input contact on the rung).

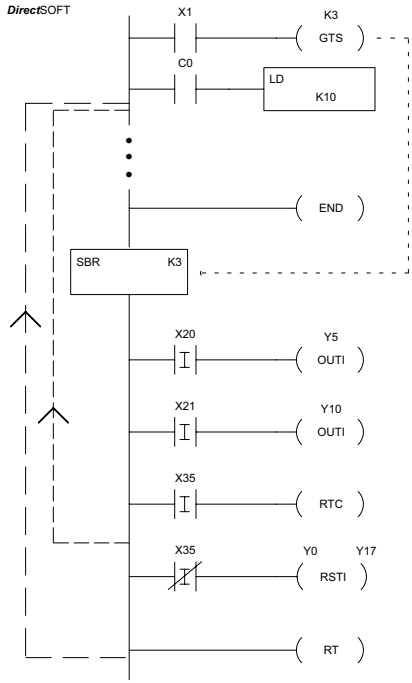
Subroutine Return Conditional (RTC)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Subroutine Return Conditional instruction is an optional instruction used with an input contact to implement a conditional return from the subroutine. The Subroutine Return (RT) is still required for termination of the Subroutine.

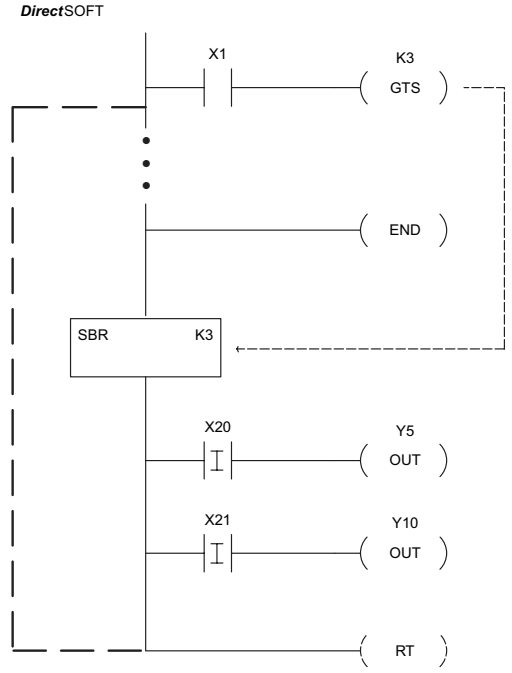
In the following example, when X1 is on, Subroutine K3 will be called. The CPU will jump to the Subroutine Label K3, and the ladder logic in the subroutine will be executed. If X35 is on, the CPU will return to the main program at the RTC instruction. If X35 is not on, Y0–Y17 will be reset to off and then the CPU will return to the main body of the program.



Handheld Programmer Keystrokes

STR	→	1	ENT					
SHFT	G	T	S	→	K	3	ENT	
•								
•								
•								
SHFT	E	N	D	ENT				
SHFT	S	SHFT	B	R	→	K	3	ENT
STR	SHFT	I	→	X	2	0	ENT	
OUT	SHFT	I	→	Y	5	ENT		
STR	SHFT	I	→	X	2	1	ENT	
OUT	SHFT	I	→	Y	1	0	ENT	
STR	SHFT	I	→	X	3	5	ENT	
SHFT	R	T	C	ENT				
STRN	SHFT	I	→	X	3	5	ENT	
RST	SHFT	I	→	Y	0	Y	1	7	ENT
SHFT	R	T	ENT					

In the following example, when X1 is on, Subroutine K3 will be called. The CPU will jump to the Subroutine Label K3 and the ladder logic in the subroutine will be executed. The CPU will return to the main body of the program after the RT instruction is executed.

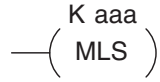


Handheld Programmer Keystrokes

$ STR	→	B 1	ENT				
SHFT	G 6	T MLR	S RST	→	D 3	ENT	
•							
•							
•							
SHFT	E 4	N TMR	D 3	ENT			
SHFT	S RST	SHFT	B 1	R ORN	→	D 3	ENT
$ STR	SHFT	I 8	→	C 2	A 0	ENT	
GX OUT	→	F 5	ENT				
$ STR	SHFT	I 8	→	C 2	B 1	ENT	
GX OUT	→	B 1	A 0	ENT			
SHFT	R ORN	T MLR	ENT				

Master Line Set (MLS)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

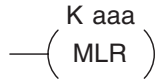


The Master Line Set instruction allows the program to control sections of ladder logic by forming a new power rail controlled by the main left power rail. The main left rail is always master line 0. When an MLS K1 instruction is used, a new power rail is created at level 1. Master Line Sets and Master Line Resets can be used to nest power rails up to seven levels deep. Note that unlike stages in RLL*PLUS*, the logic within the master control relays is still scanned and updated even though it will not function if the MLS is off.

Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
Constant	K	1-7	1-7	1-7	1-7

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

Master Line Reset (MLR)



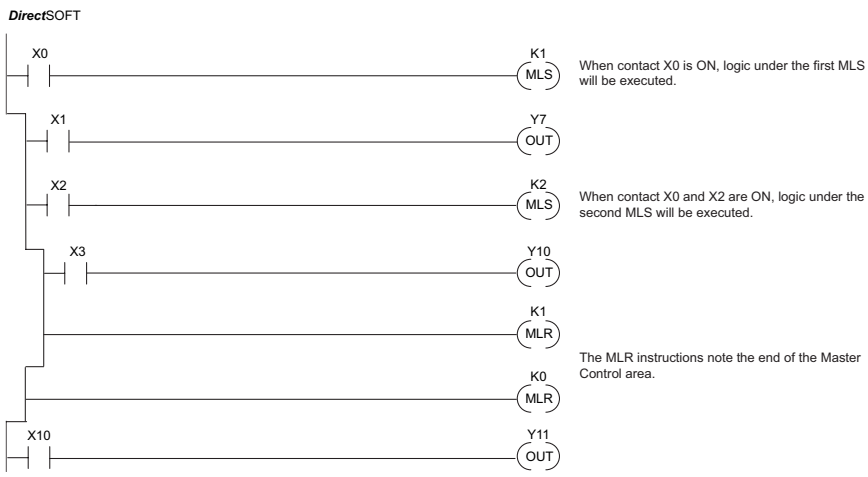
The Master Line Reset instruction marks the end of control for the corresponding MLS instruction. The MLR reference is one less than the corresponding MLS.

Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
Constant	K	1-6	1-6	1-6	1-6

Understanding Master Control Relays

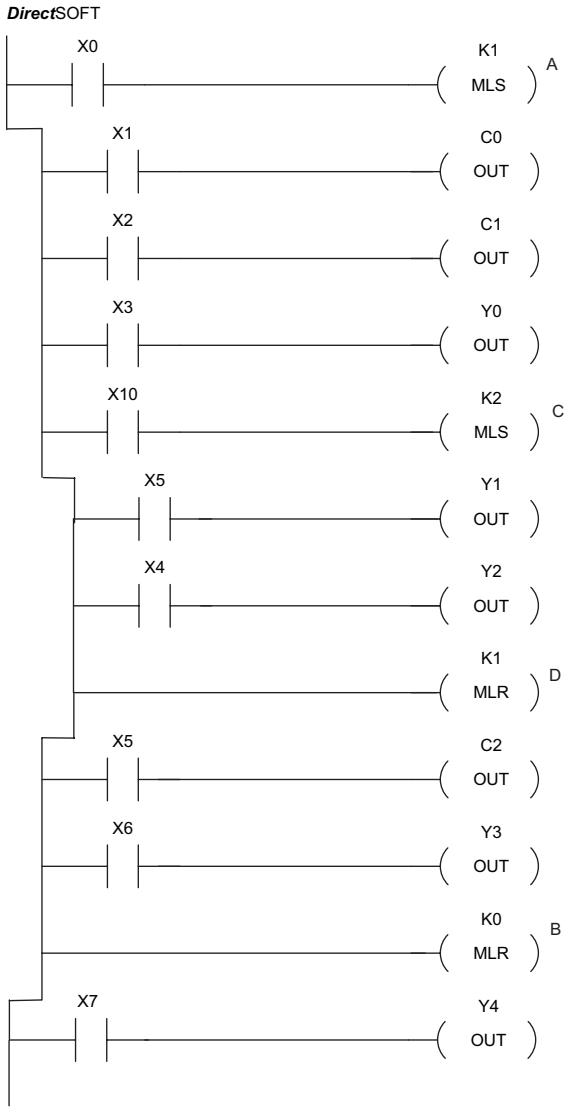
The Master Line Set (MLS) and Master Line Reset (MLR) instructions allow you to quickly enable (or disable) sections of the RLL program. This provides program control flexibility. The following example shows how the MLS and MLR instructions operate by creating a sub power rail for control logic.

DS	Used
HPP	Used



MLS/MLR Example

In the following MLS/MLR example, logic between the first MLS K1 (A) and MLR K0 (B) will function only if input X0 is on. The logic between the MLS K2 (C) and MLR K1 (D) will function only if input X10 and X0 is on. The last rung is not controlled by either of the MLS coils.



Handheld Programmer Keystrokes

$ STR	→	A 0	ENT					
Y MLS	→	B 1	ENT					
$ STR	→	B 1	ENT					
GX OUT	→	SHFT	C 2	A 0	ENT			
$ STR	→	C 2	ENT					
GX OUT	→	SHFT	C 2	B 1	ENT			
$ STR	→	D 3	ENT					
GX OUT	→	A 0	ENT					
$ STR	→	B 1	A 0	ENT				
Y MLS	→	C 2	ENT					
$ STR	→	F 5	ENT					
GX OUT	→	B 1	ENT					
$ STR	→	E 4	ENT					
GX OUT	→	C 2	ENT					
T MLR	→	B 1	ENT					
$ STR	→	F 5	ENT					
GX OUT	→	SHFT	C 2	C 2	ENT			
$ STR	→	G 6	ENT					
GX OUT	→	D 3	ENT					
T MLR	→	A 0	ENT					
$ STR	→	H 7	ENT					
GX OUT	→	E 4	ENT					

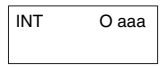
Interrupt Instructions

Interrupt (INT)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Interrupt instruction allows a section of ladder logic to be placed outside the main body of the program and executed when needed. Interrupts can be called from the program or by external interrupts via the counter interface module (D2–CTRINT), which provides 4 interrupts.



The software interrupt uses interrupt **#00 which means the hardware interrupt #0 and the software interrupt cannot be used together**.

Typically, interrupts will be used in an application where a fast response to an input is needed or a program section needs to execute faster than the normal CPU scan. The interrupt label and all associated logic must be placed after the End statement in the program. When the interrupt routine is called from the interrupt module or software interrupt, the CPU will complete execution of the instruction it is currently processing in ladder logic, then execute the designated interrupt routine. Interrupt module interrupts are labeled in octal to correspond with the hardware input signal (X1 will initiate interrupt INT1). There is only one software interrupt, and it is labeled INT 0. The program execution will continue from the point it was before the interrupt occurred once the interrupt is serviced.

The software interrupt is set up by programming the interrupt time in V7634. The valid range is 3 to 999ms. The value must be a BCD value. The interrupt will not execute if the value is out of range.

NOTE: *See the example program of a software interrupt.*

Operand Data Type		Range		
		D2-240	D2-250-1	D2-260/D2-262*
		aaa	aaa	aaa
Constant	0	0-3	0-3	0-3

D2-240/D2-250-1/D2-260/D2-262*			
Software		Hardware	
Interrupt Input	Interrupt Routine	Interrupt Input	Interrupt Routine
V7634 sets interrupt time	INT 0	X0 (cannot be used along with s/w interrupt)	INT 0
-	-	X1	INT 1
-	-	X2	INT 2
-	-	X3	INT 3

**** NOTE:*** *D2-262 supports the Software Interrupt, INT 0, only.*

Interrupt Return (IRT)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

When an Interrupt Return is executed in the interrupt routine, the CPU will return to the point in the main body of the program from which it was called. The Interrupt Return is programmed as the last instruction in an interrupt routine and is a stand alone instruction (no input contact on the rung).

Interrupt Return Conditional (IRTC)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The Interrupt Return Conditional instruction is a optional instruction used with an input contact to implement a conditional return from the interrupt routine. The Interrupt Return is required to terminate the interrupt routine.

Enable Interrupts (ENI)

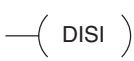
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Enable Interrupt instruction is programmed in the main body of the application program (before the End instruction) to enable hardware or software interrupts. Once the coil has been energized, interrupts will be enabled until they are disabled by the Disable Interrupt instruction.

Disable Interrupts (DISI)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Disable Interrupt instruction is programmed in the main body of the application program (before the End instruction) to disable both hardware or software interrupts. Once the coil has been energized, interrupts will be disabled until they are enabled by the Enable Interrupt instruction.

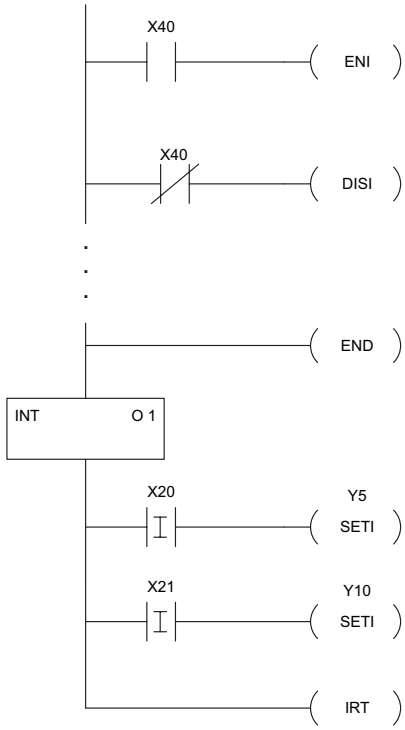


DS	Used
HPP	Used

Interrupt Example for Interrupt Module

In the following example, when X40 is on, the interrupts will be enabled. When X40 is off, the interrupts will be disabled. When an interrupt signal X1 is received, the CPU will jump to the interrupt label INT O 1. The application ladder logic in the interrupt routine will be performed. The CPU will return to the main body of the program after the IRT instruction is executed.

DirectSOFT



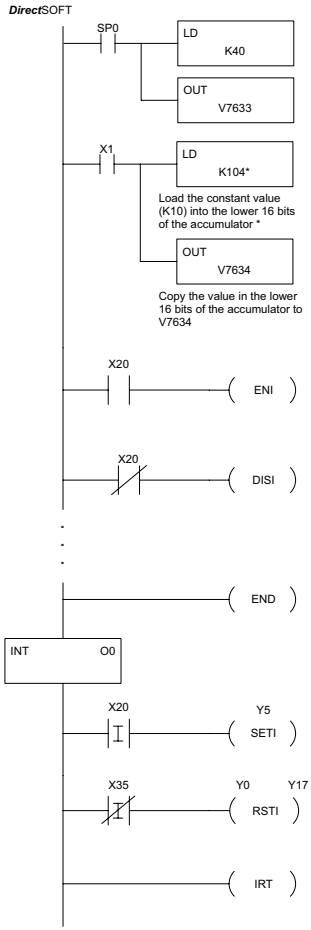
Handheld Programmer Keystrokes

$ STR	→	E 4	A 0	ENT		
SHFT	E 4	N TMR	I 8	ENT		
SP STRN	→	E 4	A 0	ENT		
SHFT	D 3	I 8	S RST	I 8	ENT	

SHFT	E 4	N TMR	D 3	ENT		
SHFT	I 8	N TMR	T MLR	→	B 1	ENT
$ STR	SHFT	I 8	→	C 2	A 0	ENT
X SET	SHFT	I 8	→	F 5	ENT	
$ STR	SHFT	I 8	→	C 2	B 1	ENT
X SET	SHFT	I 8	→	B 1	A 0	ENT
SHFT	I 8	R ORN	T MLR	ENT		

Interrupt Example for Software Interrupt

In the following example, when X1 is on, the value 10 is copied to V7634. This value sets the software interrupt to 10ms. When X20 turns on, the interrupt will be enabled. When X20 turns off, the interrupt will be disabled. Every 10ms the CPU will jump to the interrupt label INT O 0. The application ladder logic in the interrupt routine will be performed. If X35 is not on, Y0–Y17 will be reset to off and then the CPU will return to the main body of the program.



Handheld Programmer Keystrokes

$ STR	→	SHFT	SP STRN	A 0	ENT														
SHFT	L ANDST	D 3	→	SHFT	K JMP	B 4	A 0	ENT											
GX OUT	→	SHFT	V	H 7	G 6	D 3	D 3	ENT											
$ STR	→	B 1	ENT																
SHFT	L ANDST	D 3	→	SHFT	K JMP	B 1	A 0	E 4	ENT										
GX OUT	→	SHFT	V AND	H 7	G 6	D 3	E 4	ENT											
$ STR	→	C 2	A 0	ENT															
SHFT	E 4	N TMR	I 8	ENT															
SP STRN	→	C 2	A 0	ENT															
SHFT	D 3	I 8	S RST	I 8	ENT														
SHFT	E 4	N TMR	D 3	ENT															
SHFT	I 8	N TMR	T MLR	→	A 0	ENT													
$ STR	SHFT	I 8	→	C 2	A 0	ENT													
X SET	SHFT	I 8	→	F 5	ENT														
SP STRN	SHFT	I 8	→	D 3	F 5	ENT													
S RST	SHFT	I 8	→	A 0	→	B 1	H 7	ENT											
SHFT	I 8	R ORN	T MLR	ENT															

* The value entered, 3-999, must be followed by the digit 4 to complete the instruction.

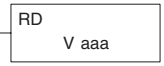
NOTE: *Only one software interrupt is allowed and it must be Int0.*

Intelligent I/O Instructions

Read from Intelligent Module (RD)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Read from Intelligent Module instruction reads a block of data (1 to 128 bytes maximum) from an intelligent I/O module into the CPU's V-memory. It loads the function parameters into the first and second level of the accumulator stack, and the accumulator by three additional instructions.



Listed below are the steps to program the Read from Intelligent module function.

DS	Used
HPP	Used

Step 1: Load the base number (0 to 3) into the first byte and the slot number (0 to 7) into the second byte of the second level of the accumulator stack.

Step 2: Load the number of bytes to be transferred into the first level of the accumulator stack (maximum of 128 bytes).

Step 3: Load the address from which the data will be read into the accumulator. This parameter must be a HEX value.

Step 4: Insert the RD instruction that specifies the starting V-memory location (Vaaa) into which the data will be read.

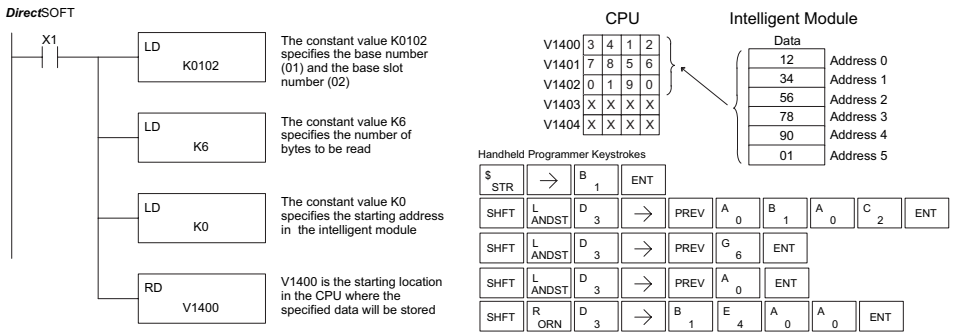
Helpful hint: Use the LDA instruction to convert an octal address to its HEX equivalent and load it into the accumulator when the hex format is required.

Operand Data Type	Range			
	D2-230	D2-240	D2-250-1	D2-260/D2-262
	aaa	aaa	aaa	aaa
V-memory V	All (See page 3-54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)

Discrete Bit Flags	Description
SP54	On when RX, WX, RD, WT instructions are executed with the wrong parameters.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

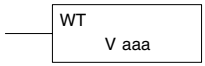
In the following example, when X1 is on, the RD instruction will read six bytes of data from an intelligent module in base 1, slot 2 starting at address 0 in the intelligent module and copy the information into V-memory locations V1400–V1402.



Write to Intelligent Module (WT)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used



The Write to Intelligent Module instruction writes a block of data (1 to 128 bytes maximum) to an intelligent I/O module from a block of V-memory in the CPU. The function parameters are loaded into the first and second level of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Read from Intelligent module function.

Step 1: Load the base number (0 to 3) into the first byte and the slot number (0 to 7) into the second byte of the second level of the accumulator stack.

Step 2: Load the number of bytes to be transferred into the first level of the accumulator stack (maximum of 128 bytes).

Step 3: Load the intelligent module address which will receive the data into the accumulator. This parameter must be a HEX value.

Step 4: Insert the WT instruction which specifies the starting V-memory location (Vaaa) where the data will be written from in the CPU.

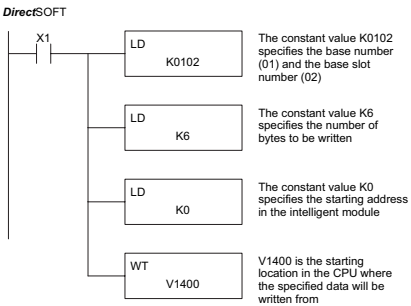
Helpful hint: Use the LDA instruction to convert an octal address to its HEX equivalent and load it into the accumulator when the hex format is required.

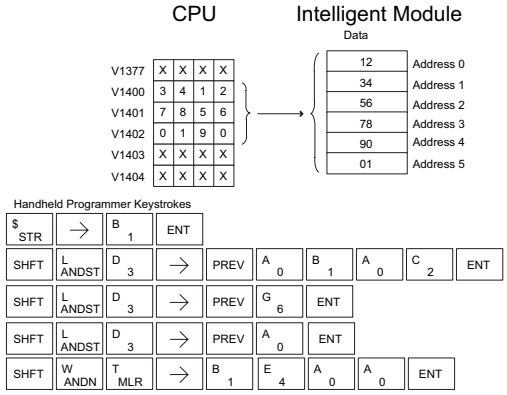
Operand Data Type	Range			
	D2-230	D2-240	D2-250-1	D2-260/D2-262
	aaa	aaa	aaa	aaa
V-memory V	All (See page 3-54)	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)

Discrete Bit Flags	Description
SP54	On when RX, WX, RD, WT instructions are executed with the wrong parameters.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the WT instruction will write six bytes of data to an intelligent module in base 1, slot 2 starting at address 0 in the intelligent module and copy the information from V-memory locations V1400–V1402.





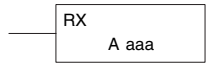
Network Instructions

Read from Network (RX)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Read from Network instruction is used by the master device on a network to read a block of data from another CPU. The function parameters are loaded into the first and second levels of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Read from Network function.



Step 1: Load the slave address (0 to 90 BCD) into the first byte, and load the PLC internal port (KF1) or slot number of the master DCM or ECOM (0 to 7) into the second byte of the second level of the accumulator stack.

Step 2: Load the number of bytes (0 to 128 BCD, multiple of 2) to be transferred into the first level of the accumulator stack.

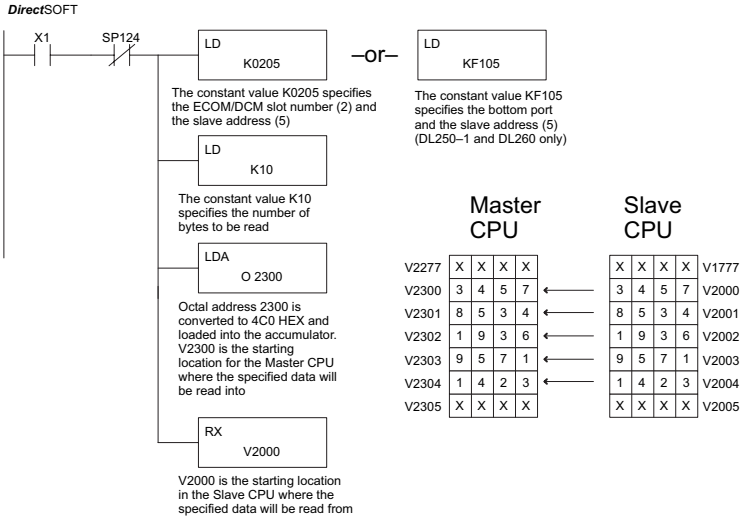
Step 3: Load the address of the data to be read into the accumulator. This parameter requires a HEX value.

Step 4: Insert the RX instruction which specifies the starting V-memory location (Aaaa) where the data will be read from in the slave.

Helpful hint: For parameters that require HEX values, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		Range		
		D2-240	D2-250-1	D2-260/D2-262
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See page 3 - 54)	All (See page 3 - 55)	All (See page 3 - 56)
Pointer	P	All V-memory (See page 3 - 54)	All V-memory (See page 3 - 55)	All V-memory (See page 3 - 56)
Inputs	X	0-477	0-777	0-1777
Outputs	Y	0-477	0-777	0-1777
Control Relays	C	0-377	0-1777	0-3777
Stage	S	0-777	0-1777	0-1777
Timer	T	0-177	0-377	0-377
Counter	CT	0-177	0-177	0-377
Global I/O	GX/GY	-	-	0-3777
Special Relay	SP	0-137 540-617	0-777	0-777

In the following example, when X1 is on and the module busy relay SP124 (see special relays) is not on, the RX instruction will access an ECOM or DCM operating as a master in slot 2. Ten consecutive bytes of data (V2000 – V2004) will be read from a CPU at station address 5 and copied into V-memory locations V2300–V2304 in the CPU with the master DCM or ECOM.



Handheld Programmer Keystrokes

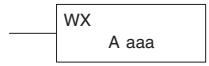
$ STR	→	B 1	ENT						
W ANDN	→	SHFT	SP STRN	B 1	C 2	E 4	ENT		
SHFT	L ANDST	D 3	→	SHFT	K JMP	C 2	A 0	F 5	ENT
SHFT	L ANDST	D 3	→	SHFT	K JMP	B 1	A 0	ENT	
SHFT	L ANDST	D 3	A 0	→	C 2	D 3	A 0	A 0	ENT
SHFT	R ORN	X SET	→	C 2	A 0	A 0	A 0	ENT	

Write to Network (WX)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Write to Network instruction is used to write a block of data from the master device to a slave device on the same network. The function parameters are loaded into the first and second levels of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Write to Network function.



Step 1: Load the slave address (0 to 90 BCD) into the first byte and the PLC internal port (KF1) or slot number of the master DCM or ECOM (0 to 7) into the second byte of the second level of the accumulator stack.

Step 2: Load the number of bytes (0 to 128 BCD, multiple of 2) to be transferred into the first level of the accumulator stack.

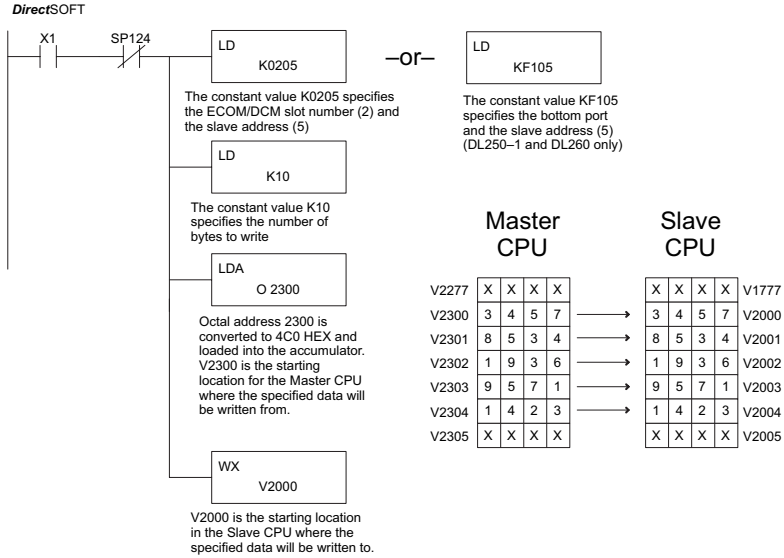
Step 3: Load the address of the data in the master that is to be written to the network into the accumulator. This parameter requires a HEX value.

Step 4: Insert the WX instruction which specifies the starting V-memory location (Aaaa) where the data will be written to the slave.

Helpful hint: — For parameters that require HEX values, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		Range		
		D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa
V-memory	V	All (See page 3 - 54)	All (See page 3 - 55)	All (See page 3 - 56)
Pointer	P	All V-memory (See page 3 - 54)	All V-memory (See page 3 - 55)	All V-memory (See page 3 - 56)
Inputs	X	0-477	0-777	0-1777
Outputs	Y	0-477	0-777	0-1777
Control Relays	C	0-377	0-1777	0-3777
Stage	S	0-777	0-1777	0-1777
Timer	T	0-177	0-377	0-377
Counter	CT	0-177	0-177	0-377
Global I/O	GX/GY	-	-	0-3777
Special Relay	SP	0-137 540-617	0-777	0-777

In the following example when X1 is on and the module busy relay SP124 (see special relays) is not on, the WX instruction will access a DCM or ECOM operating as a master in slot 2. Ten consecutive bytes of data is read from the CPU at station address 5 and copied to V-memory locations V2000–V2004 in the slave CPU.



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT							
W ANDN	→	SHFT	SP STRN	B 1	C 2	E 4	ENT			
SHFT	L ANDST	D 3	→	SHFT	K JMP	C 2	A 0	F 5	ENT	
SHFT	L ANDST	D 3	→	SHFT	K JMP	B 1	A 0	ENT		
SHFT	L ANDST	D 3	A 0	→	SHFT	O INST#	C 2	D 3	A 0	A 0	ENT
SHFT	W ANDN	X SET	→	SHFT	V AND	C 2	A 0	A 0	A 0	ENT

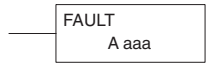
Message Instructions

Fault (FAULT)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	Used

The Fault instruction is used to display a message on the handheld programmer or *Direct*SOFT. The message has a maximum of 23 characters and can be either V-memory data, numerical constant data, or ASCII text. See Appendix G for the ASCII Conversion Table.



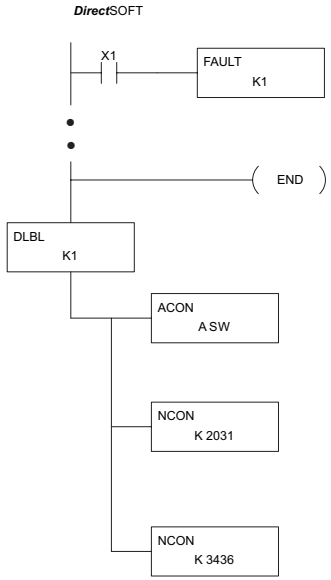
To display the value in a V-memory location, specify the V-memory location in the instruction. To display the data in ACON (ASCII constant) or NCON (Numerical constant) instructions, specify the constant (K) value for the corresponding data label area.

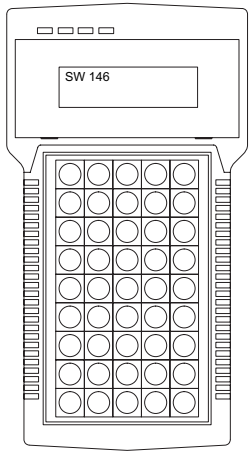
Operand Data Type		Range		
		D2-240	D2-250-1	D2-260/D2-262
	A	aaa	aaa	aaa
V-memory	V	All (See page 3-55)	All (See page 3-56)	All (See page 3-57)
Constant.	K	1-FFFF	1-FFFF	1-FFFF

NOTE: *The FAULT instruction takes a considerable amount of time to execute. This is because the FAULT parameters are stored in EEPROM. Make sure you consider the instruction execution times (shown in Appendix C) if you are attempting to use the FAULT instructions in applications that require faster than normal execution cycles.*

Fault Example

In the following example, when X1 is on, the message SW 146 will display on the handheld programmer. The NCONs use the HEX ASCII equivalent of the text to be displayed. (The HEX ASCII for a blank is 20, a 1 is 31, 4 is 34 ...)





Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	F 5	A 0	U ISG	L ANDST	T MLR	→	B 1	ENT

•
•

SHFT	E 4	N TMR	D 3	ENT						
SHFT	D 3	L ANDST	B 1	L ANDST	→	B 1	ENT			
SHFT	A 0	C 2	O INST#	N TMR	→	S RST	W ANDN	ENT		
SHFT	N TMR	C 2	O INST#	N TMR	→	C 2	A 0	D 3	B 1	ENT
SHFT	N TMR	C 2	O INST#	N TMR	→	D 3	E 4	D 3	G 6	ENT

Data Label (DLBL)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Data Label instruction marks the beginning of an ASCII/ numeric data area. DLBLs are programmed after the End statement. A maximum of 64 (D2-240 and D2-250-1D2-260/ D2-262) or 32 (D2-230) DLBL instructions can be used in a program. Multiple NCONs and ACONs can be used in a DLBL area.

DLBL
K aaa

Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
Constant	K	1-FFFF	1-FFFF	1-FFFF	1-FFFF

ASCII Constant (ACON)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The ASCII Constant instruction is used with the DLBL instruction to store ASCII text for use with other instructions. Two ASCII characters can be stored in an ACON instruction. If only one character is stored in an ACON, a leading space will be printed in the Fault message.

ACON
A aaa

Operand Data Type		D2-230 Range	D2-240 Range	D2-250-1 Range	D2-260/D2-262 Range
		aaa	aaa	aaa	aaa
ASCII	A	0-9 A-Z	0-9 A-Z	0-9 A-Z	0-9 A-Z

Numerical Constant (NCON)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Numerical Constant instruction is used with the DLBL instruction to store the HEX ASCII equivalent of numerical data for use with other instructions. Two digits can be stored in an NCON instruction.

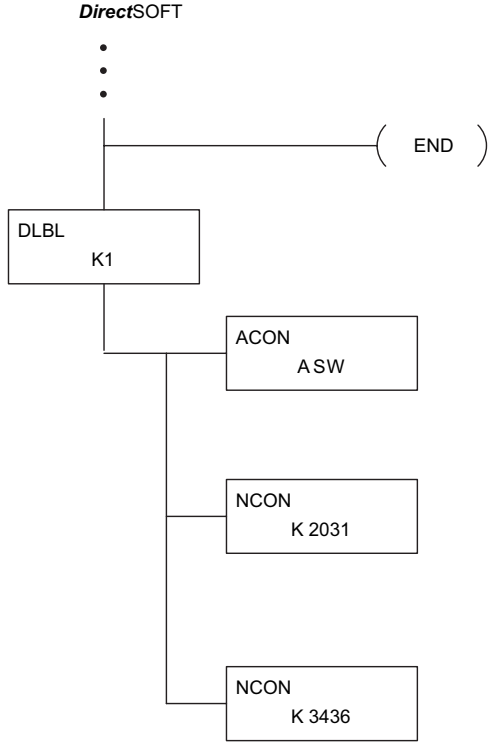
DS	Used
HPP	Used

NCON
K aaa

Operand Data Type		Range			
		D2-230	D2-240	D2-250-1	D2-260/D2-262
		aaa	aaa	aaa	aaa
Constant	K	0-FFFF	0-FFFF	0-FFFF	0-FFFF

Data Label Example

In the following example, an ACON and two NCON instructions are used within a DLBL instruction to build a text message. See the FAULT instruction for information on displaying messages.



Handheld Programmer Keystrokes

SHFT	E 4	N TMR	D 3	ENT						
SHFT	D 3	L ANDST	B 1	L ANDST	→	B 1	ENT			
SHFT	A 0	C 2	O INST#	N TMR	→	S RST	W ANDN	ENT		
SHFT	N TMR	C 2	O INST#	N TMR	→	C 2	A 0	D 3	B 1	ENT
SHFT	N TMR	C 2	O INST#	N TMR	→	D 3	E 4	D 3	G 6	ENT

Print Message (PRINT)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

The Print Message instruction prints the embedded text or text/data variable message to the specified communications port (port 2 on the D2-250-1, D2-260 and D2-262 CPUs), which must have the communications port configured.

```
PRINT     A aaa
"Hello, this is a PLC message"
```

Data Type		Range	
		D2-250-1	D2-260/D2-262
	A	aaa	aaa
Constant	K	2	2

You may recall from the CPU specifications in Chapter 3 that the D2-250–1, D2-260 and D2-262 ports are capable of several protocols. To configure a port using the Handheld Programmer, use AUX 56 and follow the prompts, making the same choices as indicated below on this page. To configure a port in ***Direct***SOFT, choose the PLC menu, then Setup, then Setup Secondary Comm Port.

- **Port:** From the port number list box at the top, choose "Port 2."
- **Protocol:** Click the check box to the left of "Non-sequence." The Setup Communication Ports dialog box opens.

- **Memory Address:** Choose a V-memory address for DirectSOFT to use to store the port setup information. You will need to reserve 66 contiguous words in V-memory for this purpose. Select "Use for printing only" if it applies.
- **Baud Rate:** Choose the baud rate that matches your printer.
- **Stop Bits, Parity:** Choose number of stop bits and parity setting to match your printer.

Then click the button indicated to send the Port 2 configuration to the CPU, and click Close. See Chapter 3 for port wiring information to connect your printer to the D2-250-1, D2-260 and D2-262 CPUs.

Port 2 on the D2-250-1, D2-260 and D2-262 CPUs has standard RS232 levels, and should work with most printer serial input connections.

Text element - used for printing character strings. The character strings are defined as the character (more than 0) ranged by the double quotation marks. Two hex numbers preceded by the dollar sign means an 8-bit ASCII character code. Also, two characters preceded by the dollar sign is interpreted according to the following table:

#	Character code	Description
1	$$	Dollar sign ($)
2	$"	Double quotation (")
3	$L or $l	Line feed (LF)
4	$N or $n	Carriage return line feed (CRLF)
5	$P or $p	Form feed
6	$R or $r	Carriage return (CR)
7	$T or $t	Tab

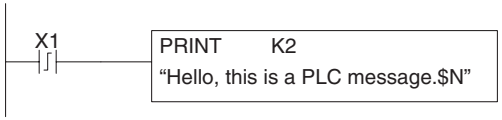
The following examples show various syntax conventions and the length of the output to the printer.

Example:

" "	Length 0 without character
"A"	Length 1 with character A
" "	Length 1 with blank
" $" "	Length 1 with double quotation mark
" $ R $ L "	Length 2 with one CR and one LF
" $ 0 D $ 0 A "	Length 2 with one CR and one LF
" $ $ "	Length 1 with one $ mark

In printing an ordinary line of text, you will need to include **double quotation** marks before and after the text string. Error code 499 will occur in the CPU when the print instruction contains invalid text or no quotations. It is important to test your PRINT instruction data during the application development.

The following example prints the message to port 2. We use a PD contact, which causes the message instruction to be active for just one scan. Note the $N at the end of the message, which produces a carriage return / line feed on the printer. This prepares the printer to print the next line, starting from the left margin.



Print the message to Port 2 when X1 makes an off-to-on transition.

V-memory element – used for printing V-memory contents in the integer format or real format. Use V-memory number or V-memory number with ":" and data type. The data types are shown in the table below. The Character code must be capital letters.

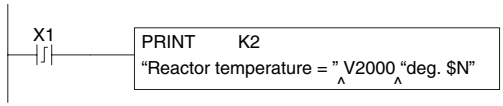
NOTE: *There must be a space entered before and after the V-memory address to separate it from the text string. Failure to do this will result in an error code 499.*

#	Character code	Description
1	none	16-bit binary (decimal number)
2	: B	4-digit BCD
3	: D	32-bit binary (decimal number)
4	: D B	8-digit BCD
5	: R	Floating point number (real number)
6	: E	Floating point number (real number with exponent)

Example:

V2000	Print binary data in V2000 for decimal number
V2000 : B	Print BCD data in V2000
V2000 : D	Print binary number in V2000 and V2001 for decimal number
V2000 : D B	Print BCD data in V2000 and V2001
V2000 : R	Print floating point number in V2000/V2001 as real number
V2000 : E	Print floating point number in V2000/V2001 as real number with exponent

Example: The following example prints a message containing text and a variable. The "reactor temperature" labels the data, which is at V2000. You can use the ': B' qualifier after the V2000 if the data is in BCD format, for example. The final string adds the units of degrees to the line of text, and the $N adds a carriage return / line feed.



the % followed by the number of characters after V-memory number for representing the text. If you assign "0" as the number of characters, the print function will read the character count from the first location. Then it will start at the next V-memory location and read that number of ASCII codes for the text from memory.

Example:

V2000 % 16	16 characters in V2000 to V2007 are printed.
V2000 % 0	The characters in V2001 to Vxxxx (determined by the number in V2000) will be printed.

Bit element – used for printing the state of the designated bit in V-memory or a relay bit. The bit element can be assigned by the designating point (.) and bit number preceded by the V-memory number or relay number. The output type is described as shown in the table below.

#	Data format	Description
1	none	Print 1 for an ON state, and 0 for an OFF state
2	: BOOL	Print "TRUE" for an ON state, and "FALSE" for an OFF state
3	: ONOFF	Print "ON" for an ON state, and "OFF" for an OFF state

Example:

V2000.15	Prints the status of bit 15 in V2000, in 1/0 format
C100	Prints the status of C100 in 1/0 format
C100 : BOOL	Prints the status of C100 in TRUE/FALSE format
C100 : ON/OFF	Prints the status of C100 in ON/OFF format
V2000.15 : BOOL	Prints the status of bit 15 in V2000 in TRUE/FALSE format

The maximum numbers of characters you can print is 128. The number of characters for each element is listed in the table below:

Element type	Maximum Characters
Text, 1 character	1
16-bit binary	6
32-bit binary	11
4-digit BCD	4
8-digit BCD	8
Floating point (real number)	13
Floating point (real with exponent)	13
V-memory/text	2
Bit (1/0 format)	1
Bit (TRUE/FALSE format)	5
Bit (ON/OFF format)	3

The Handheld Programmer's mnemonic is "PRINT," followed by the DEF field.

Special relay flags SP116 and SP117 indicate the status of the D2-250-1, D2-260 and D2-262 ports (busy, or communications error). See the appendix on special relays for a description.

NOTE: *You must use the appropriate special relay in conjunction with the PRINT command to ensure the ladder program does not try to PRINT to a port that is still busy from a previous PRINT or WX or RX instruction.*

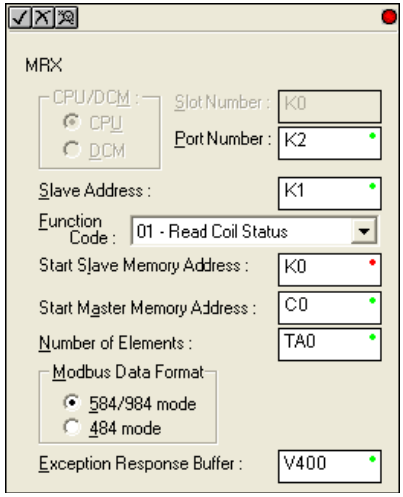
Modbus RTU Instructions (D2-260/D2-262)

Modbus Read from Network (MRX)

☒ 230
☒ 240
☒ 250-1
☑ **260**
☑ **262**

DS	Used
HPP	N/A

The Modbus Read from Network (MRX) instruction is used by the D2-260 and D2-262 network master to read a block of data from a connected slave device and to write the data into V–memory addresses within the master. The instruction allows the user to specify the Modbus Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, Modbus data format and the Exception Response Buffer.



- **Port Number:** must be D2-260/D2-262 Port 2 (K2)
- **Slave Address:** specify a slave station address (1 to 247)
- **Function Code:** The following Modbus function codes are supported by the MRX instruction:
  - 01 – Read a group of coils
  - 02 – Read a group of inputs
  - 03 – Read holding registers
  - 04 – Read input registers
  - 07 – Read Exception status
- **Start Slave Memory Address:** specifies the starting slave memory address of the data to be read. See the table on the following page.
- **Start Master Memory Address:** specifies the starting memory address in the master where the data will be placed. See the table on the following page.
- **Number of Elements:** specifies how many coils, inputs, holding registers or input registers will be read. See the table on the following page.
- **Modbus Data Format:** specifies Modbus 584/984 or 484 data format to be used.

- **Exception Response Buffer:** specifies the master memory address where the Exception Response will be placed (6 bytes in length). See the table on the following page. The exception response buffer uses 3 words. These bytes are swapped in the MRX/MWX exception response buffer V-memory so:

  V-Memory 1 Hi Byte = Function Code Byte (Most Significant Bit Set)

  V-Memory 1 Lo Byte = Address Byte

  V-Memory 2 Hi Byte = One of the CRC Bytes

  V-Memory 2 Lo Byte = Exception Code

  V-Memory 3 Hi Byte = 0

  V-Memory 3 Lo Byte = Other CRC Byte

MRX Slave Memory Address

MRX Slave Address Ranges		
Function Code	**Modbus Data Format**	**Slave Address Range(s)**
01-Read Coil	484 Mode	1-999
01-Read Coil	584/984 Mode	1-65535
02-Read Input Status	484 Mode	1001-1999
02-Read Input Status	584/984 Mode	10001-19999 (5 digit) or 100001-165535 (6 digit)
03-Read Holding Register	484 Mode	4001-4999
03-Read Holding Register	584/984	40001-49999 9 (5 digit) or 4000001-465535 (6 digit)
04-Read Input Register	484 Mode	3001-3999
04-Read Input Register	584/984 Mode	30001-39999 (5 digit) or 3000001-365535 (6 digit)
07-Read Exception Status	484 and 584/984 Mode	n/a

MRX Master Memory Addresses

MRX Master Memory Address Ranges		
Operand Data Type		**D2-260/D2-262 Range**
Inputs	X	0-1777
Outputs	Y	0-1777
Control Relays	C	0-3777
Stage Bits	S	0-1777
Timer Bits	T	0-377
Counter Bits	CT	0-377
Special Relays	SP	0-777
V-memory	V	all (see page 3-57)
Global Inputs	GX	0-3777
Global Outputs	GY	0-3777

MRX Number of Elements

Number of Elements	
Operand Data Type	**D2-260/D2-262 Range**
V-memory V	All (see page 3-57)
Constant K	Bits: 1-2000 Registers: 1-125

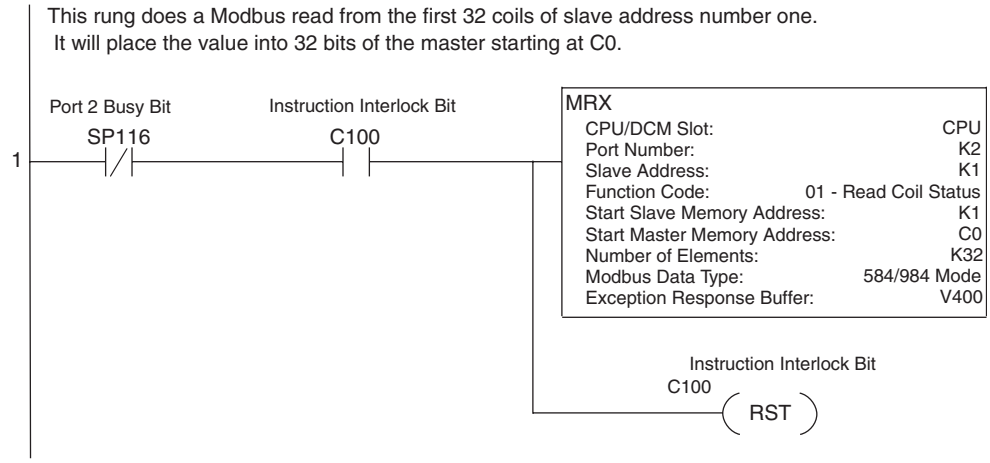
MRX Exception Response Buffer

Exception Response Buffer	
Operand Data Type	**D2-260/D2-262 Range**
V-memory V	All (see page 3-57)

MRX Example

D2-260 and D2-262 port 2 has two Special Relay contacts associated with it (see Appendix D for comm port special relays). One indicates "Port busy" (SP116), and the other indicates "Port Communication Error" (SP117). The "Port Busy" bit is on while the PLC communicates with the slave. When the bit is off, the program can initiate the next network request. The "Port Communication Error" bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an MRX or MWX instruction is executed.

Typically, network communications will last longer than one CPU scan. The program must wait for the communications to finish before starting the next transaction.

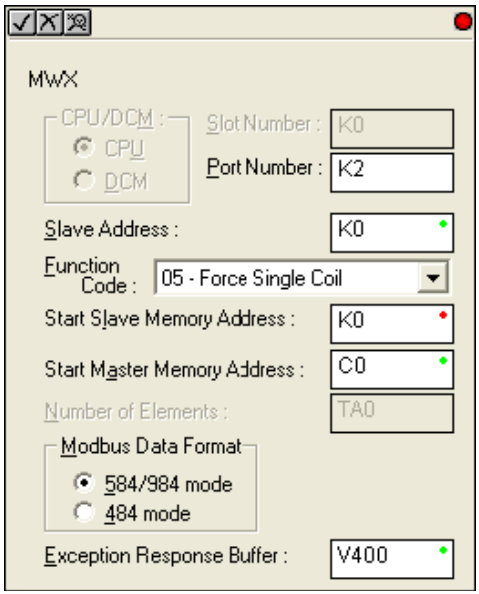


Modbus Write to Network (MWX)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

The Modbus Write to Network (MWX) instruction is used by a D2-262 or D2-262 network master to write a block of data from V-memory to Modbus memory addresses within a slave device on the network. The instruction allows the user to specify the Modbus Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, Modbus data format and the Exception Response Buffer.



- **Port Number:** must be D2-260 or D2-262 Port 2 (K2)
- **Slave Address:** specify a slave station address (0 to 247)
- **Function Code:** The following Modbus function codes are supported by the MWX instruction:
  - 05 – Force Single coil
  - 06 – Preset Single Register
  - 15 – Force Multiple Coils
  - 16 – Preset Multiple Registers
- **Start Slave Memory Address:** specifies the starting slave memory address where the data will be written
- **Start Master Memory Address:** specifies the starting address of the data in the master that is to be written to the slave
- **Number of Elements:** specifies how many consecutive coils or registers will be written to. This field is only active when either function code 15 or 16 is selected
- **Modbus Data Format:** specifies Modbus 584/984 or 484 data format to be used

- **Exception Response Buffer:** specifies the master memory address where the Exception Response will be placed (6-bytes in length). See the table on the following page. The exception response buffer uses 3 words. These bytes are swapped in the MRX/MWX exception response buffer V-memory so:

  V-Memory 1 Hi Byte = Function Code Byte (Most Significant Bit Set)

  V-Memory 1 Lo Byte = Address Byte

  V-Memory 2 Hi Byte = One of the CRC Bytes

  V-Memory 2 Lo Byte = Exception Code

  V-Memory 3 Hi Byte = 0

  V-Memory 3 Lo Byte = Other CRC Byte

MWX Slave Memory Address

MWX Slave Address Ranges		
Function Code	**Modbus Data Format**	**Slave Address Range(s)**
05 - Force Sinlge Coil	484 Mode	1-999
05 - Force Single Coil	584/984 Mode	1-65535
06 - Preset Single Register	484 Mode	4001-4999
06 - Preset Single Register	584/984 Mode	40001-49999 (5 digit) or 400001-465535 (6 digit)
15 - Force Multiple Coils	484 Mode	1-999
15 - Force Multiple Coils	584/984 Mode	1-65535
16 - Preset Multiple Registers	484 Mode	4001-4999
16 - Preset Multiple Registers	584/984 Mode	40001-49999 (5 digit) or 4000001-465535 (6 digit)

MWX Master Memory Addresses

MWX Master Memory Address Ranges		
Operand Data Type		**D2-260/D2-262 Range**
Inputs	X	0-1777
Outputs	Y	0-1777
Control Relays	C	0-3777
Stage Bits	S	0-1777
Timer Bits	T	0-377
Counter Bits	CT	0-377
Special Relays	SP	0-777
V-memory	V	all (see page 3-57)
Global Inputs	GX	0-3777
Global Outputs	GY	0-3777

MWX Number of Elements

Number of Elements		
Operand Data Type		D2-260/D2-262 Range
V-memory	V	all (see page 3-57)
Constant	K	Bits: 1-2000 Registers: 1-125

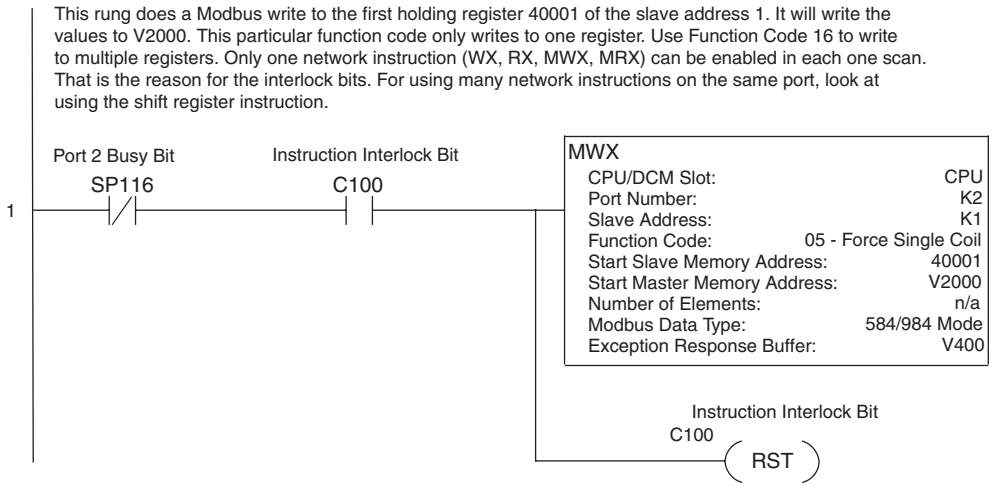
MWX Exception Response Buffer

Exception Response Buffer		
Operand Data Type		D2-260/D2-262 Range
V-memory	V	all (see page 3-57)

MWX Example

D2-260 and D2-262 port 2 has two Special Relay contacts associated with it (see Appendix D for comm port special relays). One indicates "Port busy" (SP116), and the other indicates "Port Communication Error" (SP117). The "Port Busy" bit is on while the PLC communicates with the slave. When the bit is off, the program can initiate the next network request. The "Port Communication Error" bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an MRX or MWX instruction is executed.

Typically, network communications will last longer than one CPU scan. The program must wait for the communications to finish before starting the next transaction.



ASCII Instructions (D2-260/D2-262)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

The D2-260 and D2-262 CPU supports several instructions and methods that allow ASCII strings to be read into and written from the PLC communications ports.

Specifically, port 2 on the D2-260 and D2-262 can be used for either reading or writing raw ASCII strings, but cannot be used for both on the same CPU.

The D2-260 and D2-262 can also decipher ASCII embedded within a supported protocol (K–Sequence, ***Direct***NET, Modbus, Ethernet) via the CPU ports, H2–ECOM or D2–DCM module.

ASCII character tables and descriptions can be found at www.asciitable.com.

Reading ASCII Input Strings

There are several methods which the D2-260 and D2-262 can use to read ASCII input strings:

1) **ASCII IN (AIN)** – This instruction configures port 2 for raw ASCII input strings with parameters such as fixed and variable length ASCII strings, termination characters, byte swapping options, and instruction control bits. Use barcode scanners, weight scales, etc., to write raw ASCII input strings into port 2 based on the (AIN) instruction's parameters.
2) Write embedded ASCII strings directly to V–memory from an external HMI or similar master device via a supported communications protocol using the CPU ports, H2–ECOM or D2–DCM module. The AIN instruction is not used in this case.
3) If a D2-260 or a D2-262 PLC is a master on a network, the Network Read instruction (RX) can be used to read embedded ASCII data from a slave device via a supported communications protocol using port 2, H2–ECOM or D2–DCM module. The RX instruction places the data directly into V–memory.

Writing ASCII Output Strings

The following instructions can be used to write ASCII output strings:

1) Print from V–memory (PRINTV) – Use this instruction to write raw ASCII strings out of port 2 to a display panel or a serial printer, etc. The instruction features the starting V–memory address, string length, byte swapping options, etc. When the instruction's permissive bit is enabled, the string is written to port 2.
2) Print to V–memory (VPRINT) – Use this instruction to create pre–coded ASCII strings in the PLC (i.e. alarm messages). When the instruction's permissive bit is enabled, the message is loaded into a pre–defined V–memory address location. Then use the PRINTV instruction to write the pre–coded ASCII string out of port 2. American, European and Asian Time/Date stamps are supported.

Additionally, if a D2-260 or a D2-262 PLC is a master on a network, the Network Write instruction (WX) can be used to write embedded ASCII data to an HMI or slave device directly from V–memory via a supported communications protocol using port 2, H2–ECOM or D2–DCM module.

Managing the ASCII Strings

The following instructions can be helpful in managing the ASCII strings within the CPU's V-memory:

ASCII Find (AFIND) – Finds where a specific portion of the ASCII string is located in continuous V-memory addresses. Forward and reverse searches are supported.

ASCII Extract (AEX) – Extracts a specific portion (usually some data value) from the ASCII find location or other known ASCII data location.

Compare V-memory (CMPV) – This instruction is used to compare two blocks of V-memory addresses and is usually used to detect a change in an ASCII string. Compared data types must be of the same format (i.e., BCD, ASCII, etc.).

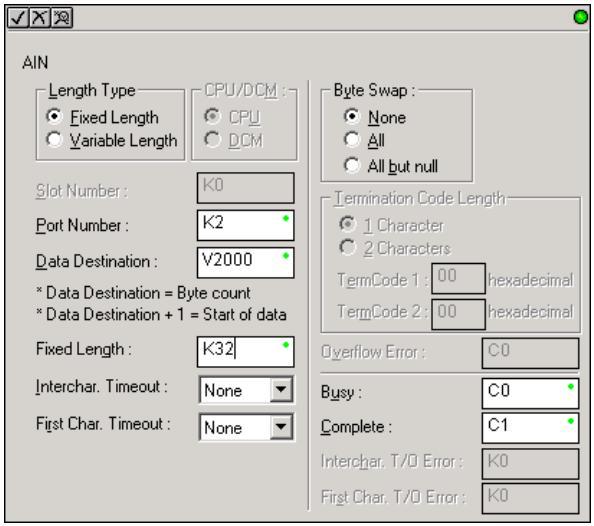
Swap Bytes (SWAPB) – usually used to swap V-memory bytes on ASCII data that was written directly to V-memory from an external HMI or similar master device via a communications protocol. The AIN and AEX instructions have a built–in byte swap feature.

ASCII Input (AIN)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

The ASCII Input instruction allows the CPU to receive ASCII strings through the specified communications port and places the string into a series of specified V-memory registers. The ASCII data can be received as a fixed number of bytes or as a variable length string with a specified termination character(s). Other features include Byte Swap preferences, Character Timeout, and user-defined flag bits for Busy, Complete and Timeout Error.



AIN Fixed Length Configuration

- **Length Type**: select fixed length based on the length of the ASCII string that will be sent to the CPU port.
- **Port Number**: must be D2-260/D2-262 port 2 (K2).
- **Data Destination**: specifies where the ASCII string will be placed in V–memory.
- **Fixed Length**: specifies the length, in bytes, of the fixed-length ASCII string the port will receive.
- **Inter–character Timeout**: if the amount of time between incoming ASCII characters exceeds the set time, the specified Timeout Error bit will be set. No data will be stored at the Data Destination V–memory location. The bit will reset when the AIN instruction permissive bits are disabled. *None* selection disables this feature.
- **First Character Timeout**: if the amount of time from when the AIN is enabled to the time the first character is received exceeds the set time, the specified First Character Timeout bit will be set. The bit will reset when the AIN instruction permissive bits are disabled. *None* selection disables this feature.
- **Byte Swap**: swaps the high–byte and low–byte within each V–memory register of the Fixed Length ASCII string. See the SWAPB instruction for details.
- **Busy Bit**: is ON while the AIN instruction is receiving ASCII data.
- **Complete Bit**: is set once the ASCII data has been received for the specified fixed length and reset when the AIN instruction permissive bits are disabled.
- **Inter–character Timeout Error Bit**: is set when the Character Timeout is exceed. See Character Timeout explanation above.
- **First Character Timeout Error Bit**: is set when the First Character Timeout is exceeded. See First Character Timeout explanation above.

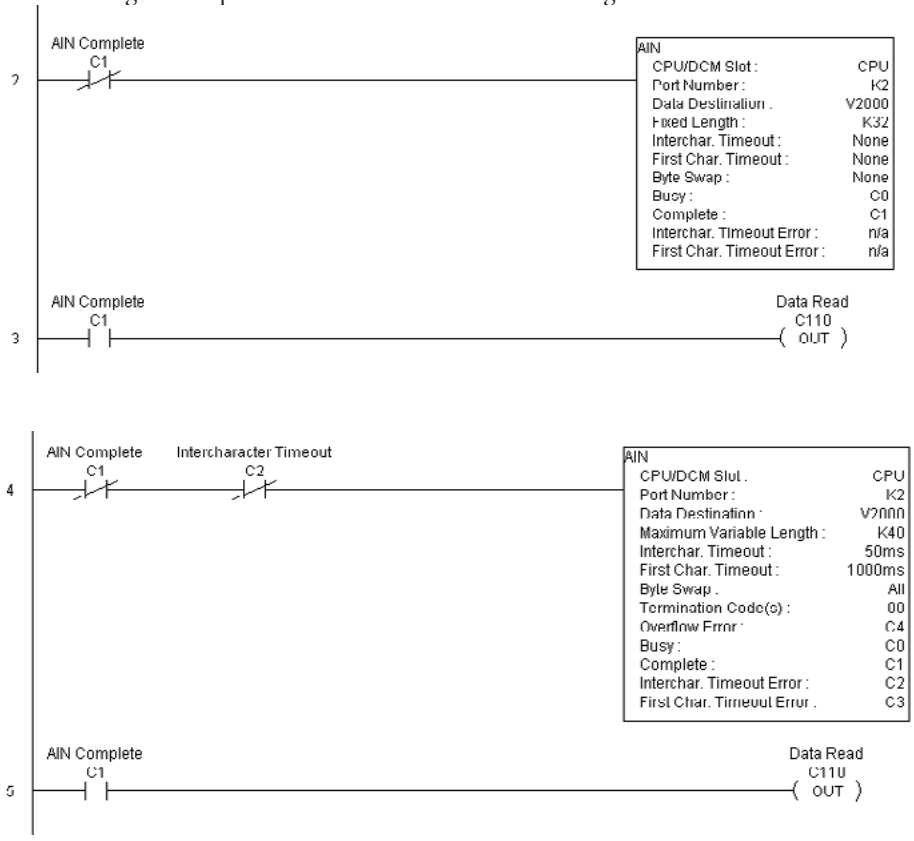
Parameter	D2-260/D2-262 Range
Data Destination	All V-memory (See page 3 -56)
Fixed Length	K1-128
Bits: Busy, Complete, Timeout Error, Overflow	C0-3777

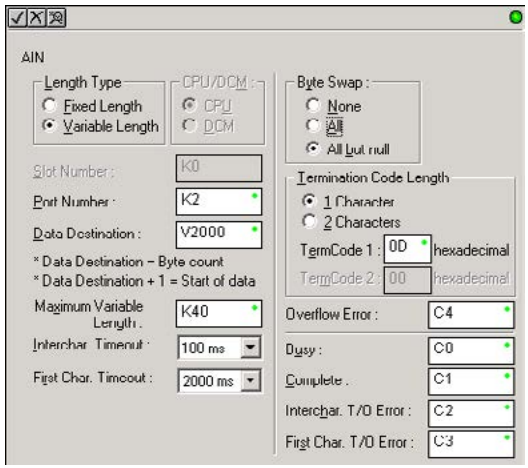
Discrete Bit Flags	Description
SP53	On if the CPU cannot execute the instruction
SP71	On when a value used by the instruction is invalid
SP116	On when CPU port 2 is communicating with another device
SP117	On when CPU port 2 has experienced a communication error

AIN Fixed Length Examples

Fixed Length example when the PLC is reading the port continuously and timing is not critical.

Fixed Length example when character to character timing is critical.





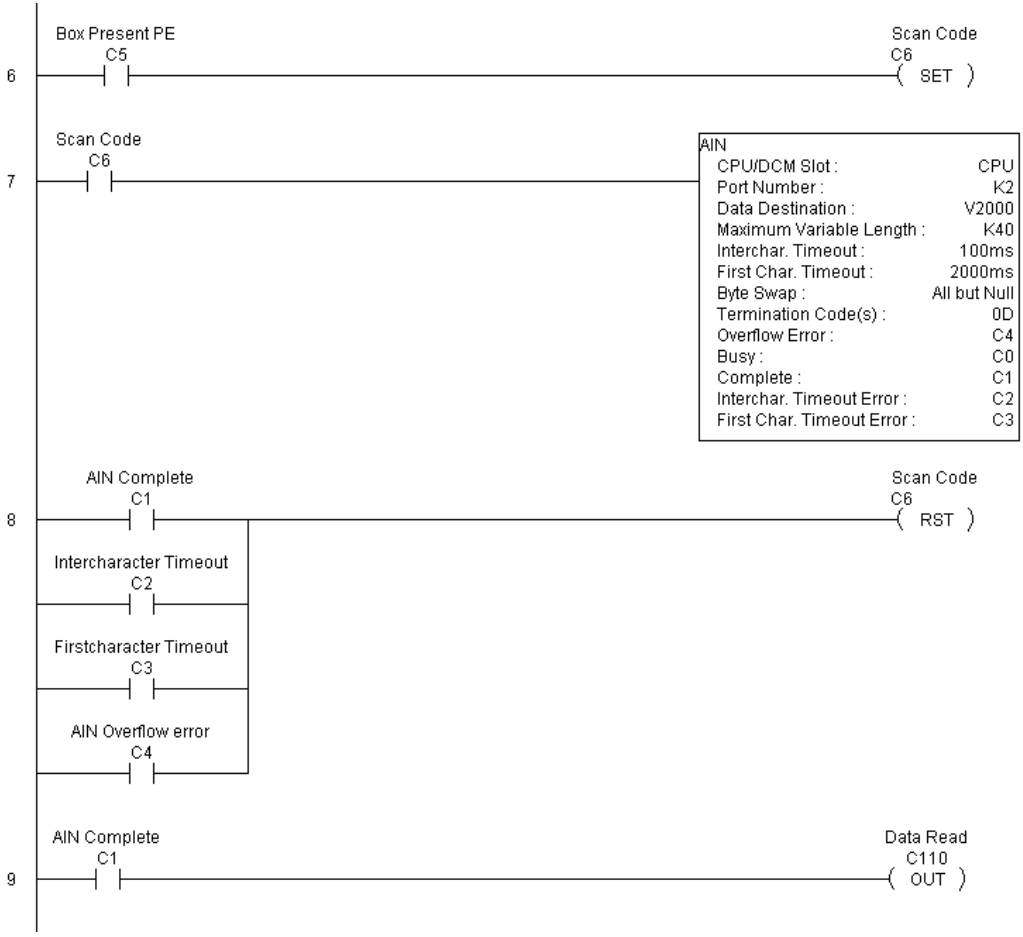
AIN Variable Length Configuration:

- **Length Type**: select Variable Length if the ASCII string length followed by termination characters will vary in length.
- **Port Number**: must be D2-260/D2-262 port 2 (K2).
- **Data Destination**: specifies where the ASCII string will be placed in V–memory.
- **Maximum Variable Length**: specifies, in bytes, the maximum length of a Variable Length ASCII string the port will receive.
- **Inter–character Timeout**: if the amount of time between incoming ASCII characters exceeds the set time, the Timeout Error bit will be set. No data will be stored at the Data Destination V–memory location. The Timeout Error bit will reset when the AIN instruction permissive bits are disabled. *None* selection disables this feature.
- **First Character Timeout**: if the amount of time from when the AIN is enabled to the time the first character is received exceeds the set time, the specified First Character Timeout bit will be set. The bit will reset when the AIN instruction permissive bits are disabled. *None* selection disables this feature.
- **Byte Swap**: swaps the high–byte and low–byte within each V–memory register of the Varaible Length ASCII string. See the SWAPB instruction for details.
- **Termination Code Length**: consists of either 1 or 2 characters. Refer to the ASCII table in Appendix G.
- **Overflow Error Bit**: is set when the ASCII data received exceeds the Maximum Variable Length specified.
- **Busy Bit**: is ON while the AIN instruction is receiving ASCII data.
- **Complete Bit**: is set once the ASCII data has been received up to the termination code characters. It will be reset when the AIN instruction permissive bits are disabled.
- **Inter–character Timeout Error Bit**: is set when the Character Timeout is exceed. See Character Timeout explanation above.
- **First Character Timeout Error Bit**: is set when the First Character Timeout is exceeded. See First Character Timeout explanation above.

Parameter	D2-260/D2-262 Range
Data Destination	All V-memory (See page 3-57)
Max. Variable Length	K1-128
Bits: Busy, Complete, Timeout Error, Overflow	C0-3777

AIN Variable Length Example

AIN Variable Length example used to read barcodes on boxes (PE = photoelectric sensor).

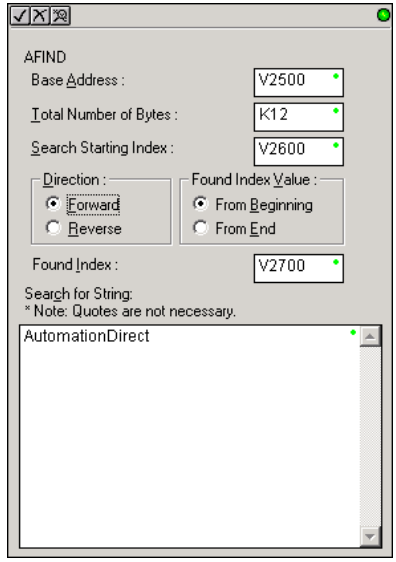


ASCII Find (AFIND)

☒ 230
☒ 240
☒ 250-1
☑ **260**
☑ **262**

DS	Used
HPP	N/A

The ASCII Find instruction locates a specific ASCII string or portion of an ASCII string within a range of V-memory registers and places the string's Found Index number (byte number where desired string is found) in Hex, into a specified V-memory register. Other features include, Search Starting Index number for skipping over unnecessary bytes before beginning the FIND operation, Forward or Reverse direction search, and From Beginning and From End selections to reference the Found Index Value.



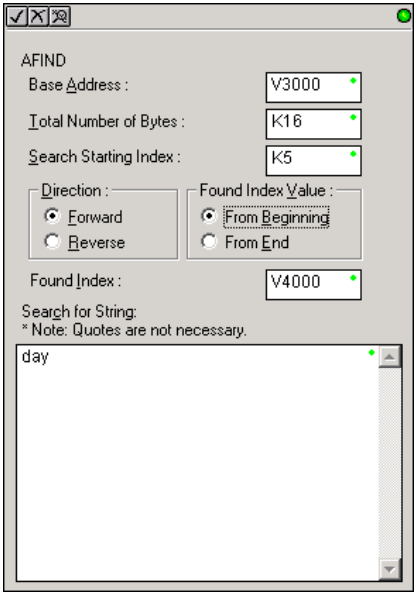
- **Base Address**: specifies the beginning V-memory register where the entire ASCII string is stored in memory.
- **Total Number of Bytes**: specifies the total number of bytes to search for the desired ASCII string.
- **Search Starting Index**: specifies which byte to skip to (with respect to the Base Address) before beginning the search.
- **Direction**: Forward begins the search from lower numbered V-memory registers to higher numbered V-memory registers. Reverse does the search from higher numbered V–memory registers to lower-numbered V-memory registers.
- **Found Index Value**: specifies whether the Beginning or the End byte of the ASCII string found will be loaded into the Found Index register.
- **Found Index**: specifies the V–memory register where the Found Index Value will be stored. A value of FFFF will result if the desired string is not located in the memory registers specified. A value of EEEE will result if there is a conflict in the AFIND search parameters specified.
- **Search for String**: up to 128 characters.

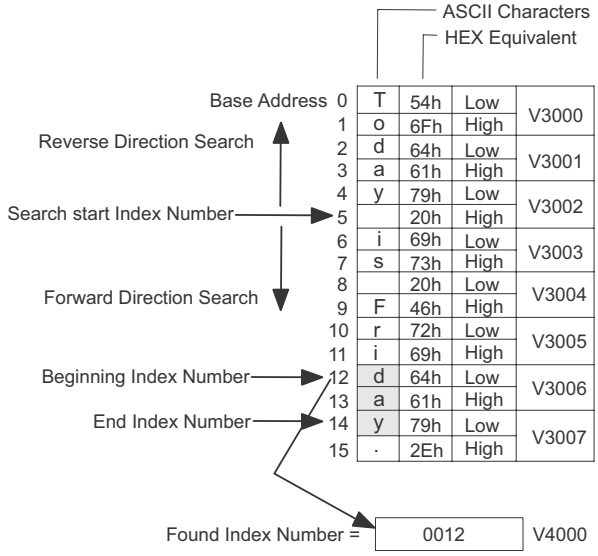
Parameter	D2-260/D2-262 Range
Base Address	All V-memory (See page 3-57)
Total Number of Bytes	All V-memory (See page 3-57) or K1-128
Search Starting Index	All V-memory (See page 3-57) or K0-127
Found Index	All V-memory (See page 3-57)

Discrete Bit Flags	Description
SP53	On if the CPU cannot execute the instruction.
SP71	On when a value used by the instruction is invalid.

AFIND Search Example

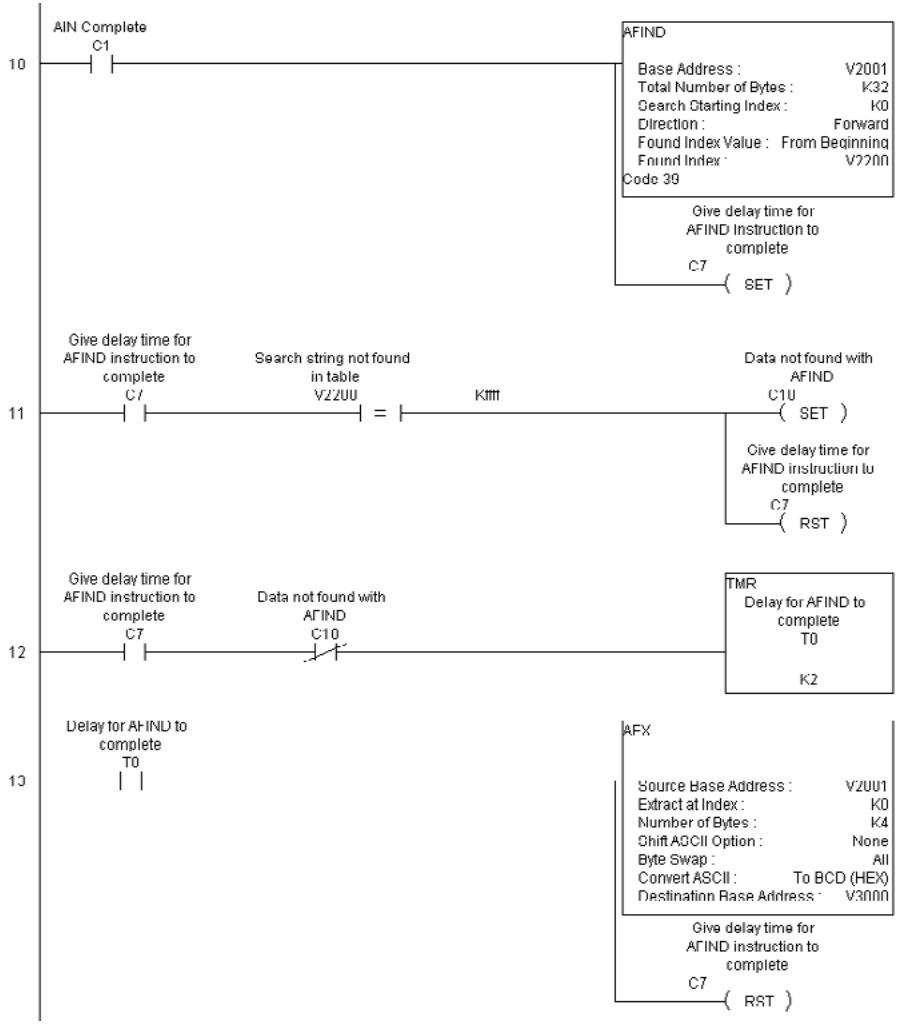
In the following example, the AFIND instruction is used to search for the "day" portion of "Friday" in the ASCII string "Today is Friday," which had previously been loaded into V–memory. Note that a Search Starting Index of constant (K) 5 combined with a Forward Direction Search is used to prevent finding the "day" portion of the word "Today." The Found Index will be placed into V4000.





AFIND Example Combined with AEX Instruction

When an AIN instruction has executed, its Complete bit can be used to trigger an AFIND instruction to search for a desired portion of the ASCII string. Once the string is found, the AEX instruction can be used to extract the located string.



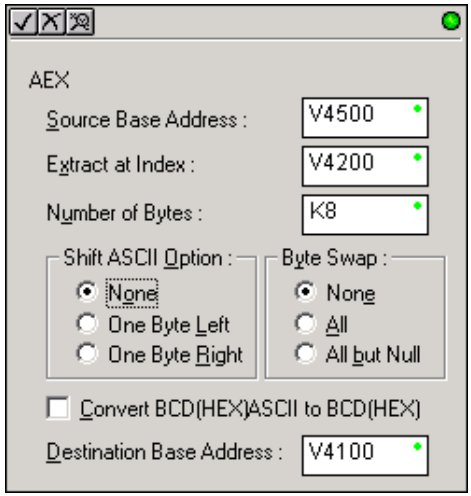
ASCII Extract (AEX)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

The ASCII Extract instruction extracts a specified number of bytes of ASCII data from one series of V-memory registers and places them into another series of V-memory registers. Other features include Extract at Index for skipping over unnecessary bytes before begining the Extract operation, Shift ASCII Option, for One Byte Left or One Byte Right, Byte Swap and Convert data to a BCD format number.

- **Source Base Address**: specifies the beginning V-memory register where the entire ASCII string is stored in memory.
- **Extract at Index**: specifies which byte to skip to (with respect to the Source Base Address) before extracting the data.
- **Number of Bytes**: specifies the number of bytes to be extracted.
- **Shift ASCII Option**: shifts all extracted data one byte left or one byte right to displace "unwanted" characters, if necessary.
- **Byte Swap**: swaps the high–byte and the low–byte within each V-memory register of the extracted data. See the SWAPB instruction for details.
- **Convert BCD(Hex) ASCII to BCD (Hex)**: if enabled, this will convert ASCII numerical characters to Hexadecimal numerical values.
- **Destination Base Address**: specifies the V-memory register where the extracted data will be stored.



Parameter	D2-260/D2-262 Range
Source Base Address	All V-memory (See page 3-57)
Extract at Index	All V-memory (See page 3-57) or K0-127
Number of Bytes	K1-128
Destination Base Address	All V-memory (See page 3-57)

Discrete Bit Flags	Description
SP53	On if the CPU cannot execute the instruction.
SP71	On when a value used by the instruction is invalid.

See the previous page for an example using the AEX instruction.

ASCII Compare (CMPV)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

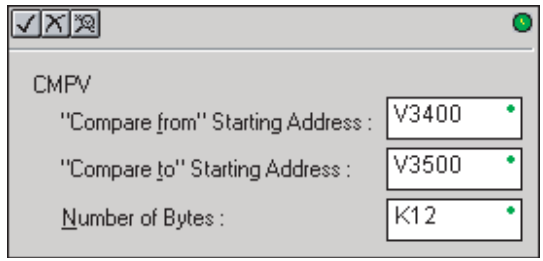
DS	Used
HPP	N/A

The ASCII Compare instruction compares two groups of V–memory registers. The CMPV will compare any data type (ASCII to ASCII, BCD to BCD, etc) of one series (group) of V–memory registers to another series of V–memory registers for a specified byte length.

"Compare from" Starting Address: specifies the beginning V–memory register of the first group of V–memory registers to be compared from.

"Compare to" Starting Address: specifies the beginning V–memory register of the second group of V–memory registers to be compared to.

Number of Bytes: specifies the length of each V–memory group to be compared.



SP61 = 1 (ON), the result is equal

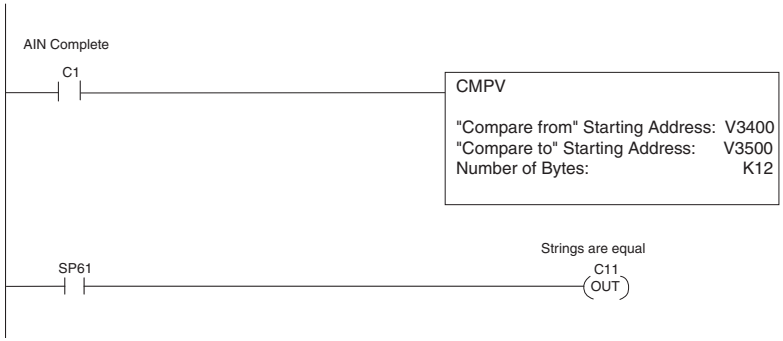
SP61 = 0 (OFF), the result is not equal

Parameter	D2-260/D2-262 Range
Compare from Starting Address	All V-memory (See page 3-57)
Compare to Starting Address	All V-memory (See page 3-57)
Number of Bytes	All V-memory (See page 3-57) or K0-127

Discrete Bit Flags	Description
SP53	On if the CPU cannot execute the instruction.
SP61	On when result is equal.
SP71	On when a value used by the instruction is invalid.

CMPV Example

The CMPV instruction executes when the AIN instruction is complete. If the compared V–memory tables are equal, SP61 will turn ON.



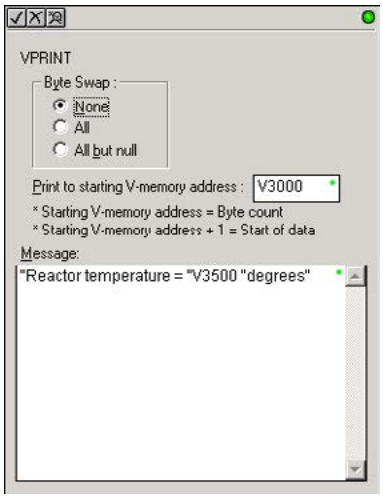
ASCII Print to V-memory (VPRINT)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

The ASCII Print to V–memory instruction will write a specified ASCII string into a series of V–memory registers. Other features include Byte Swap, options to suppress or convert leading zeros or spaces, and _Date and _Time options for U.S., European, and Asian date formats and 12- or 24-hour time formats.

- **Byte Swap**: swaps the high–byte and low–byte within each V–memory register to which the ASCII string is printed. See the SWAPB instruction for details.
- **Print to Starting V–memory Address**: specifies the beginning of a series of V–memory addresses where the ASCII string will be placed by the VPRINT instruction.
- **Starting V–memory Address**: the first V–memory register of the series of registers specified will contain the ASCII string's length in bytes.
- **Starting V–memory Address +1**: the 2nd and subsequent registers will contain the ASCII string printed to V–memory.



Parameter	D2-260/D2-262 Range
Print to Starting V-memory Address	All V-memory (See page 3-57)

Discrete Bit Flags	Description
SP53	On if the CPU cannot execute the instruction.
SP71	On when a value used by the instruction is invalid.

VPRINT Time/Date Stamping

The codes in the table below can be used in the VPRINT ASCII string message to "print to V–memory" the current time and/or date.

#	Character Code	Date/Time Stamp Options
1	_Date:us	American standard (month/day/2 digit year)
2	_Date:e	European standard (day/month/2 digit year)
3	_Date:a	Asian standard (2 digit year/month/day)
4	_Time:12	standard 12 hour clock (0-12 hour:min am/pm)
5	_Time:24	standard 24 hour clock (0-23 hour:min am/pm)

VPRINT V-memory element

The following modifiers can be used in the VPRINT ASCII string message to "print to V–memory" register contents in integer format or real format. Use V-memory number or V-memory number with ":" and data type. The data types are shown in the table below. The Character code must be capital letters.

NOTE: *There must be a space entered before and after the V-memory address to separate it from the text string. Failure to do this will result in error code 499.*

#	Character Code	Description
1	none	16-bit binary (decimal number)
2	: B	4-digit BCD
3	: D	32-bit binary (decimal number)
4	: D B	8-digit BCD
5	: R	Floating point number (real number)
6	: E	Floating point number (real number with exponent)

Examples:

V2000 Print binary data in V2000 for decimal number

V2000 : B Print BCD data in V2000

V2000 : D Print binary number in V2000 and V2001 for decimal number

V2000 : D B Print BCD data in V2000 and V2001

V2000 : R Print floating point number in V2000/V2001 as real number

V2000 : E Print floating point number in V2000/V2001 as real number with exponent

The following modifiers can be added to any of the modifies above to suppress or convert leading zeros or spaces. The character code must be capital letters.

#	Character Code	Description
1	S	Suppresses leading spaces
2	C0	Converts leading spaces to zeros
3	0	Suppresses leading zeros

Example with V2000 = 0018 (binary format)

V-memory Register with Modifier	Number of Characters			
	1	2	3	4
V2000	0	0	1	8
V2000:B	0	0	1	2
V2000:B0	1	2		

Example with V2000 = sp sp18 (binary format) where sp = space

V-memory Register with Modifier	Number of Characters			
	1	2	3	4
V2000	sp	sp	1	8
V2000:B	sp	sp	1	2
V2000:BS	1	2		
V2000:BCD	0	0	1	2

VPRINT V-memory text element

The following is used for "printing to V-memory" text stored in registers. Use the % followed by the number of characters after V-memory number for representing the text. If you assign "0" as the number of characters, the function will read the character count from the first location. Then it will start at the next V-memory location and read that number of ASCII codes for the text from memory.

Example:

V2000 % 16	16 characters in V2000 to V2007 are printed.
V2000 % 0	The characters in V2001 to Vxxxx (determined by the number in V2000) will be printed.

VPRINT Bit element

The following is used for "printing to V–memory" the state of the designated bit in V-memory or a control relay bit. The bit element can be assigned by the designating point (.) and bit number preceded by the V-memory number or relay number. The output type is described as shown in the table below.

#	Data format	Description
1	none	Print 1 for an ON state, and 0 for an OFF state
2	: BOOL	Print "TRUE" for an ON state, and "FALSE" for an OFF state
3	: ONOFF	Print "ON" for an ON state, and "OFF" for an OFF state

Example:

V2000.15	Prints the status of bit 15 in V2000, in 1/0 format
C100	Prints the status of C100 in 1/0 format
C100 : BOOL	Prints the status of C100 in TRUE/FALSE format
C100 : ON/OFF	Prints the status of C100 in ON/OFF format
V2000.15 : BOOL	Prints the status of bit 15 in V2000 in TRUE/FALSE format

The maximum numbers of characters you can VPRINT is 128. The number of characters required for each element, regardless of whether the :S, :C0 or :0 modifiers are used, is listed in the table below.

Element Type	Maximum Characters
Text, 1 character	1
16-bit binary	6
32-bit binary	11
4-digit BCD	4
8-digit BCD	8
Floating point (real number)	13
Floating point (real with exponent)	13
V-memory/text	2
Bit (1/0 format)	1
Bit (TRUE/FALSE format)	5
Bit (ON/OFF format)	3

Text element

The following is used for "printing to V-memory" character strings. The character strings are defined as the character (more than 0) ranged by the double quotation marks. Two hex numbers preceded by the dollar sign means an 8-bit ASCII character code. Also, two characters preceded by the dollar sign is interpreted according to the following table:

#	Character code	Description
1	$$	Dollar sign ($)
2	$"	Double quotation (")
3	$L or $l	Line feed (LF)
4	$N or $n	Carriage return line feed (CRLF)
5	$P or $p	Form feed
6	$R or $r	Carriage return (CR)
7	$T or $t	Tab

The following examples show various syntax conventions and the length of the output to the printer.

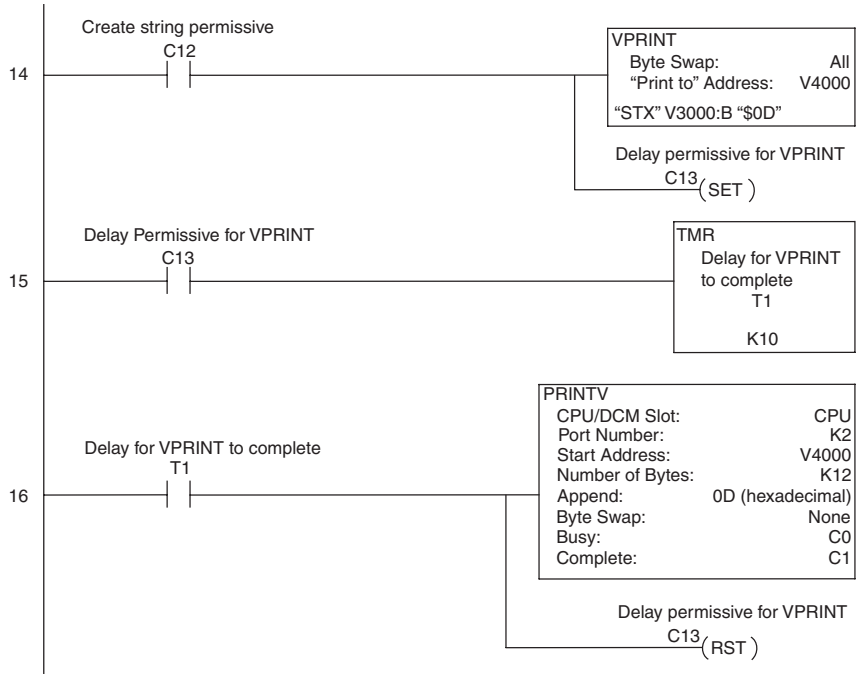
Example:

” ”	Length 0 without character
”A”	Length 1 with character A
” ”	Length 1 with blank
” $” ”	Length 1 with double quotation mark
” $ R $ L ”	Length 2 with one CR and one LF
” $ 0 D $ 0 A ”	Length 2 with one CR and one LF
” $ $ ”	Length 1 with one $ mark

In printing an ordinary line of text, you will need to include **double quotation** marks before and after the text string. Error code 499 will occur in the CPU when the print instruction contains invalid text or no quotations. It is important to test your VPRINT instruction data during application development.

VPRINT Example Combined with PRINTV Instruction

The VPRINT instruction is used to create a string in V–memory. The PRINTV is used to print the string out of port 2.



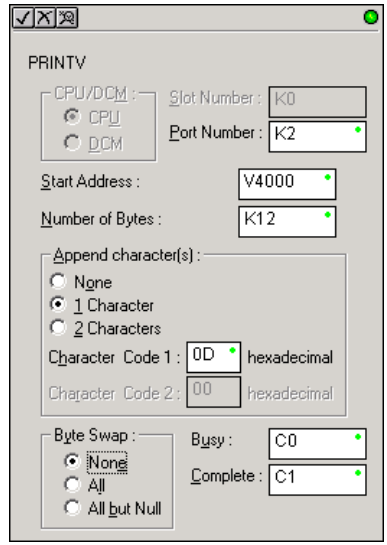
ASCII Print from V-memory (PRINTV)

☒ 230
☒ 240
☒ 250-1
☑ **260**
☑ **262**

DS	Used
HPP	N/A

The ASCII Print from V–memory instruction will send an ASCII string out of the designated communications port from a specified series of V–memory registers for a specified length in number of bytes. Other features include user specified Append Characters to be placed after the desired data string for devices that require specific termination character(s), Byte Swap options, and user specified flags for Busy and Complete.

- **Port Number**: must be D2-260/D2-262 port 2 (K2).
- **Start Address**: specifies the beginning of series of V–memory registers that contain the ASCII string to print.
- **Number of Bytes**: specifies the length of the string to print.
- **Append Characters**: specifies ASCII characters to be added to the end of the string for devices that require specific termination characters.
- **Byte Swap**: swaps the high–byte and low–byte within each V–memory register of the string while printing. See the SWAPB instruction for details.
- **Busy Bit**: will be ON while the instruction is printing ASCII data.
- **Complete Bit**: will be set once the ASCII data has been printed and reset when the PRINTV instruction permissive bits are disabled.



Parameter	D2-260/D2-262 Range
Port Number	port 2 (K2)
Start Address	All V-memory (See page 3-57)
Number of Bytes	All V-memory (See page 3-57) or K1-128
Bits: Busy, Complete	C0-3777

Discrete Bit Flags	Description
SP53	On if the CPU cannot execute the instruction.
SP71	On when a value used by the instruction is invalid.
SP116	On when CPU port 2 is communicating with another device.
SP117	On when CPU port 2 has experienced a communication error.

See the facing page for an example using the PRINTV instruction.

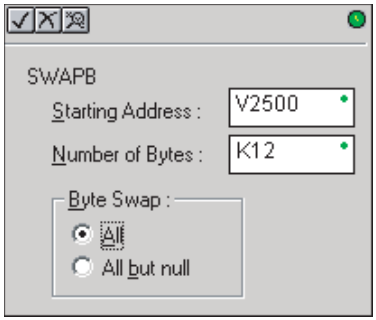
ASCII Swap Bytes (SWAPB)

☒ 230
☒ 240
☒ 250-1
☑ 260
☑ 262

DS	Used
HPP	N/A

The ASCII Swap Bytes instruction swaps byte positions (high–byte to low–byte and low–byte to high–byte) within each V-memory register of a series of V-memory registers for a specified number of bytes.

- **Starting Address**: specifies the beginning of a series of V–memory registers the instruction will use to begin byte swapping
- **Number of Bytes**: specifies the number of bytes, beginning with the Starting Address, to byte swap

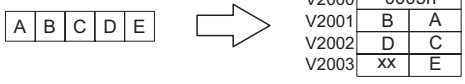


Parameter	D2-260/D2-262 Range
Starting Address	All V-memory (See page 3-57)
Number of Bytes	All V-memory (See page 3-57) or K1 to 128

Discrete Bit Flags	Description
SP53	On if the CPU cannot execute the instruction.
SP71	On when a value used by the instruction is invalid.

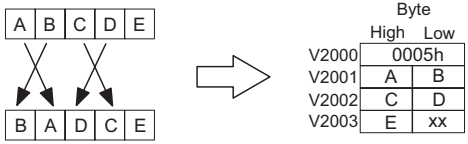
Byte Swap Preferences

No Byte Swapping (AIN, AEX, PRINTV, VPRINT)



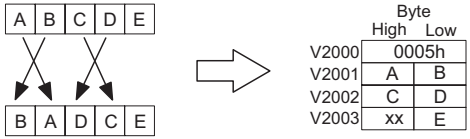
	Byte High	Byte Low
V2000	0005h	
V2001	B	A
V2002	D	C
V2003	xx	E

Byte Swap All



	Byte High	Byte Low
V2000	0005h	
V2001	A	B
V2002	C	D
V2003	E	xx

Byte Swap All but Null



	Byte High	Byte Low
V2000	0005h	
V2001	A	B
V2002	C	D
V2003	xx	E

SWAPB Example

The AIN Complete bit is used to trigger the SWAPB instruction. Use a one–shot so the SWAPB only executes once.

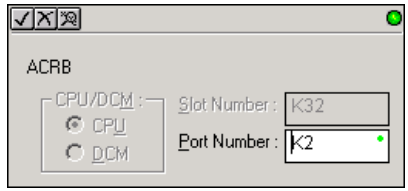
ASCII Clear Buffer (ACRB)

☒ 230
☒ 240
☒ 250-1
☑ **260**
☑ **262**

DS	Used
HPP	N/A

The ASCII Clear Buffer instruction will clear the ASCII receive buffer of the specified communications port number.

Port Number: must be D2-260/D2-262 port 2 (K2)



ACRB Example

The AIN Complete bit or the AIN diagnostic bits are used to clear the ASCII buffer.

Intelligent Box (IBox) Instructions (D2-250-1, D2-260 and D2-262 Only)

A new class of instructions, called Ibox Instructions, became available with the introduction of *Direct*SOFT. These powerful, yet easy-to-use instructions simplify many of the more complicated tasks that could previously be accomplished only through the use of multiple RLL Instructions. The IBox Instructions are supported by D2-250–1, D2-260 and D2-262 PLCs. The D2-250-1 CPU requires firmware version v4.60 or later, while the D2-260 and D2-262 CPUs requires firmware version v1.0 or later. For more information on *Direct*SOFT or to download our free version, please visit our Web site at: **www.automationdirect.com**.

Analog Helper IBoxes		
Instruction	**IBox #**	**Page**
Analog Input / Output Combo Module Pointer Setup (ANLGCMB)	IB-462	5-233
Analog Input Module Pointer Setup (ANLGIN)	IB-460	5-235
Analog Output Module Pointer Setup (ANLGOUT)	IB-461	5-237
Analog Scale 12-Bit BCD to BCD (ANSCL)	IB-423	5-239
Analog Scale 12-Bit Binary to Binary (ANSCLB)	IB-403	5-240
Filter Over Time - BCD (FILTER)	IB-422	5-241
Filter Over Time - Binary (FILTERB)	IB-402	5-243
Hi/Low Alarm - BCD (HILOAL)	IB-421	5-245
Hi/Low Alarm - Binary (HILOALB)	IB-401	5-247

Discrete Helper IBoxes		
Instruction	**Ibox #**	**Page**
Off Delay Timer (OFFDTMR)	IB-302	5-249
On Delay Timer (ONDTMR)	IB-301	5-251
One Shot (ONESHOT)	IB-303	5-253
Push On / Push Off Circuit (PONOFF)	IB-300	5-254

Memory IBoxes		
Instruction	**Ibox #**	**Page**
Move Single Word (MOVEW)	IB-200	5-255
Move Double Word (MOVED)	IB-201	5-256

NOTE: *Check your CPU firmware version using* ***Direct****SOFT: PLC Menu > Diagnostics > System Information. The latest firmware and update tool are available from:*

http://support.automationdirect.com/firmware/index.html

Intelligent Box (IBox) Instructions, Continued (D2-250-1, D2-260 and D2-262 Only)

Math IBoxes		
Instruction	**Ibox #**	**Page**
BCD to Real with Implied Decimal Point (BCDTOR)	IB-560	5-257
Double BCD to Real with Implied Decimal Point (BCDTORD)	IB-562	5-258
Math - BCD (MATHBCD)	IB-521	5-259
Math - Binary (MATHBIN)	IB-501	5-261
Math - Real (MATHR)	IB-541	5-263
Real to BCD with Implied Decimal Point and Rounding (RTOBCD)	IB-561	5-264
Real to Double BCD with Implied Decimal Point and Rounding (RTOBCDD)	IB-563	5-265
Square BCD (SQUARE)	IB-523	5-266
Square Binary (SQUAREB)	IB-503	5-267
Square Real(SQUARER)	IB-543	5-268
Sum BCD Numbers (SUMBCD)	IB-522	5-269
Sum Binary Numbers (SUMBIN)	IB-502	5-270
Sum Real Numbers (SUMR)	IB-542	5-271

Communication IBoxes		
Instruction	**Ibox #**	**Page**
ECOM100 Configuration (ECOM100)	IB-710	5-273
ECOM100 Disable DHCP (ECDHCPD)	IB-736	5-275
ECOM100 Enable DHCP (ECDHCPE)	IB-735	5-277
ECOM100 Query DHCP Setting (ECDHCPQ)	IB-734	5-279
ECOM100 Send E-mail (ECEMAIL)	IB-711	5-281
ECOM100 Restore Default E-mail Setup (ECEMRDS)	IB-713	5-287
ECOM100 E-mail Setup (ECEMSUP)	IB-712	5-290
ECOM100 IP Setup (ECIPSUP)	IB-717	5-294
ECOM100 Read Description (ECRDDES)	IB-726	5-296
ECOM100 Read Gateway Address (ECRDGWA)	IB-730	5-298
ECOM100 Read IP Address (ECRDIP)	IB-722	5-300
ECOM100 Read Module ID (ECRDMID)	IB-720	5-302
ECOM100 Read Module Name (ECRDNAM)	IB-724	5-304
ECOM100 Read Subnet Mask (ECRDSNM)	IB-732	5-306
ECOM100 Write Description (ECWRDES)	IB-727	5-308
ECOM100 Write Gateway Address (ECWRGWA)	IB-731	5-310
ECOM100 Write IP Address (ECWRIP)	IB-723	5-312
ECOM100 Write Module ID (ECWRMID)	IB-721	5-314
ECOM100 Write Name (ECWRNAM)	IB-725	5-316
ECOM100 Write Subnet Mask (ECWRSNM)	IB-733	5-318
ECOM100 RX Network Read (ECRX)	IB-740	5-320
ECOM100 WX Network Write (ECWX)	IB-741	5-323
NETCFG Network Configuration (NETCFG)	IB-700	5-326
Network RX Read (NETRX)	IB-701	5-328
Network WX Write (NETWX)	IB-702	5-331

Intelligent Box (IBox) Instructions, Continued (D2-250-1, D2-260 and D2-262 Only)

Counter I/O IBoxes (Work with H2-CTRIO and H2-CTRIO2)		
Instruction	**Ibox #**	**Page**
CTRIO Configuration (CTRIO)	IB-1000	5-334
CTRIO Add Entry to End of Preset Table (CTRADPT)	IB-1005	5-336
CTRIO Clear Preset Table (CTRCLRT)	IB-1007	5-339
CTRIO Edit Preset Table Entry (CTREDPT)	IB-1003	5-342
CTRIO Edit Preset Table Entry and Reload (CTREDRL)	IB-1002	5-346
CTRIO Initialize Preset Table (CTRINPT)	IB-1004	5-350
CTRIO Initialize Preset Table (CTRINTR)	IB-1010	5-354
CTRIO Load Profile (CTRLDPR)	IB-1001	5-358
CTRIO Read Error (CTRRDER)	IB-1014	5-361
CTRIO Run to Limit Mode (CTRRTLM)	IB-1011	5-363
CTRIO Run to Position Mode (CTRRTPM)	IB-1012	5-366
CTRIO Velocity Mode (CTRVELO)	IB-1013	5-369
CTRIO Write File to ROM (CTRWFTR)	IB-1006	5-372

NOTE: *Check your CPU firmware version using* ***Direct****SOFT: PLC Menu > Diagnostics > System Information. The latest firmware and update tool are available from:*

http://support.automationdirect.com/firmware/index.html

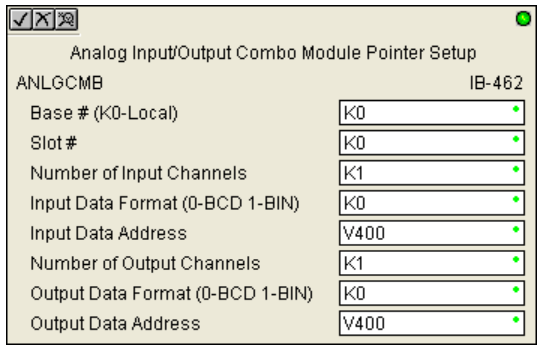
Analog Input/Output Combo Module Pointer Setup (ANLGCMB) (IB-462)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

The Analog Input/Output Combo Module Pointer Setup instruction generates the logic to configure the pointer method for an analog input/output combination module on the first PLC scan following a Program to Run transition.

The ANLGCMB IBox instruction determines the data format and Pointer addresses based on the CPU type, the Base# and the module Slot#.



The Input Data Address is the starting location in user V-memory where the analog input data values will be stored, one location for each input channel enabled.

The Output Data Address is the starting location in user V-memory where the analog output data values will be stored by ladder code or external device, one location for each output channel enabled.

Since the IBox logic only executes on the first scan, the instruction cannot have any input logic.

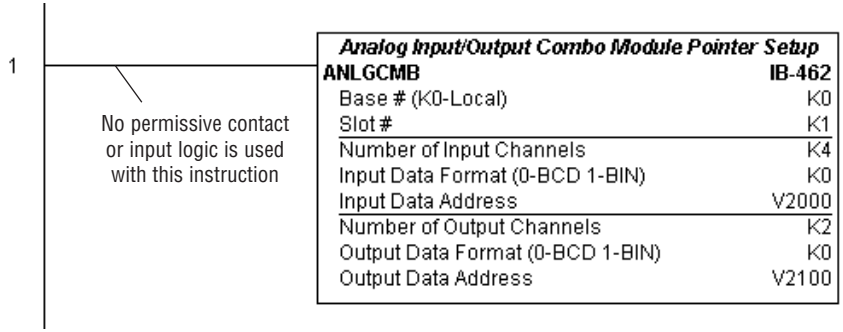
ANLGCMB Parameters

- Base # (K0-Local): specifies which base the module is in.
- Slot #: specifies which slot is occupied by the analog module.
- Number of Input Channels: specifies the number of analog input channels to scan.
- Input Data Format (0-BCD 1-BIN): specifies the analog input data format (BCD or Binary) - the binary format may be used for displaying data on some OI panels.
- Input Data Address: specifies the starting V-memory location that will be used to store the analog input data.
- Number of Output Channels: specifies the number of analog output channels that will be used.
- Output Data Format (0-BCD 1-BIN): specifies the format of the analog output data (BCD or Binary).
- Output Data Address: specifies the starting V-memory location that will be used to source the analog output data.

Parameter		DL205 Range
Base # (K0-Local)	K	K0-3
Slot #	K	K0-7
Number of Input Channels	K	K1-8
Input Data Format (0-BCD 1-BIN)	K	BCD: K0; Binary: K1
Input Data Address	V	See DL205 V-memory map - Data Words
Number of Output Channels	K	K1-8
Output Data Format (0-BCD 1-BIN)	K	BCD: K0; Binary: K1
Output Data Address	V	See DL205 V-memory map - Data Words

ANLGCMB Example

In the following example, the ANLGCMB instruction is used to set up the pointer method for an analog I/O combination module that is installed in option slot 2. Four input channels are enabled and the analog data will be written to V2000 - V2003 in BCD format. Two output channels are enabled and the analog values will be read from V2100 - V2101 in BCD format.



NOTE**: An Analog I/O IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

Analog Input Module Pointer Setup (ANLGIN) (IB-460)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

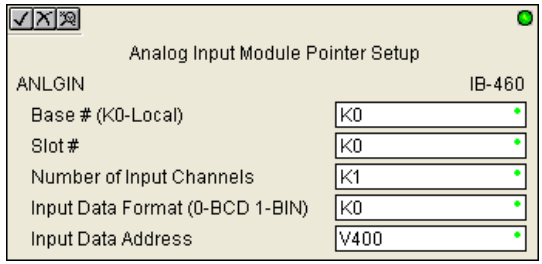
DS5	Used
HPP	N/A

Analog Input Module Pointer Setup generates the logic to configure the pointer method for one analog input module on the first PLC scan following a Program to Run transition.

This IBox determines the data format and Pointer addresses based on the CPU type, the Base#, and the Slot#.

The Input Data Address is the starting location in user V-memory where the analog input data values will be stored, one location for each input channel enabled.

Since this logic only executes on the first scan, this IBox cannot have any input logic.



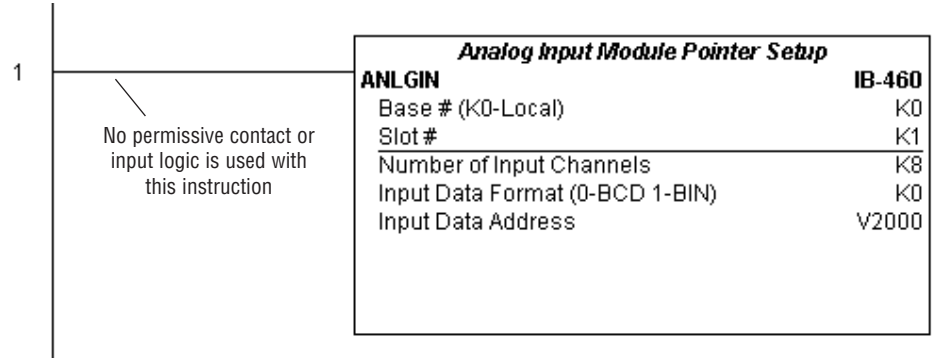
ANLGIN Parameters

- Base # (K0-Local): specifies which base the analog module is in.
- Slot #: specifies which PLC slot is occupied by the analog module.
- Number of Input Channels: specifies the number of input channels to scan.
- Input Data Format (0-BCD 1-BIN): specifies the analog input data format (BCD or Binary) - the binary format may be used for displaying data on some OI panels.
- Input Data Address: specifies the starting V-memory location that will be used to store the analog input data.

Parameter		DL205 Range
Base # (K0-Local)	K	K0-3
Slot #	K	K0-7
Number of Input Channels	K	K1-8
Input Data Format (0-BCD 1-BIN)	K	BCD: K0; Binary: K1
Input Data Address	V	See DL205 V-memory map - Data Words

ANLGIN Example

In the following example, the ANLGIN instruction is used to set up the pointer method for an analog input module that is installed in option slot 1. Eight input channels are enabled and the analog data will be written to V2000 - V2007 in BCD format.



***NOTE**: An Analog I/O IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Analog Output Module Pointer Setup (ANLGOUT) (IB-461)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

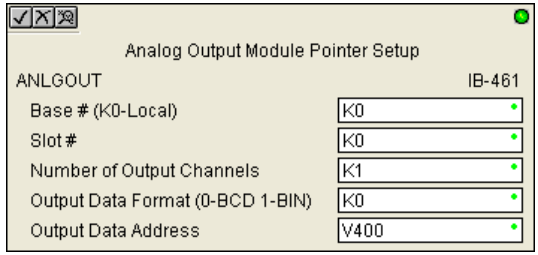
DS5	Used
HPP	N/A

Analog Output Module Pointer Setup generates the logic to configure the pointer method for one analog output module on the first PLC scan following a Program to Run transition.

This IBox determines the data format and Pointer addresses based on the CPU type, the Base#, and the Slot#.

The Output Data Address is the starting location in user V-memory where the analog output data values will be placed by ladder code or external device, one location for each output channel enabled.

Since this logic only executes on the first scan, this IBox cannot have any input logic.



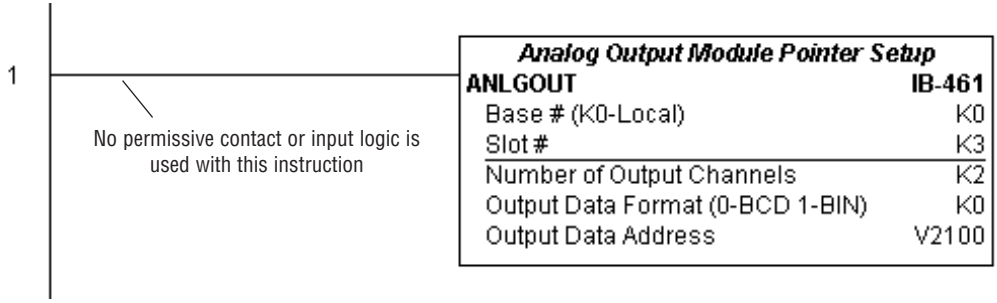
ANLGOUT Parameters

- Base # (K0-Local): specifies which base the analog module is in
- Slot #: specifies which PLC slot is occupied by the analog module
- Number of Output Channels: specifies the number of analog output channels that will be used
- Output Data Format (0-BCD 1-BIN): specifies the format of the analog output data (BCD or Binary)
- Output Data Address: specifies the starting V-memory location that will be used to source the analog output data

Parameter		DL205 Range
Base # (K0-Local)	K	K0-3
Slot #	K	K0-7
Number of Output Channels	K	K1-8
Output Data Format (0-BCD 1-BIN)	K	BCD: K0; Binary: K1
Output Data Address	V	See DL205 V-memory map - Data Words

ANLGOUT Example

In the following example, the ANLGOUT instruction is used to set up the pointer method for an analog output module that is installed in option slot 3. Two output channels are enabled and the analog data will be read from V2100 - V2101 in BCD format.



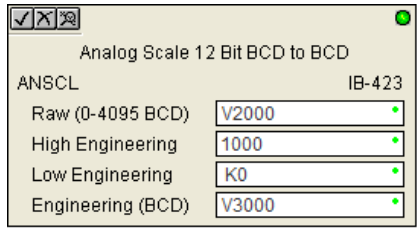
***NOTE**: An Analog I/O IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Analog Scale 12-Bit BCD to BCD (ANSCL) (IB-423)

☒ 230
☒ 240
☑ **250-1**
☑ **260**
☑ **262**

DS5	Used
HPP	N/A

Analog Scale 12-Bit BCD to BCD scales a 12-bit BCD analog value (0 to 4095 BCD) into BCD engineering units. You specify the engineering unit high value (when raw is 4095), and the engineering low value (when raw is 0), and the output V-memory address where you want to place the scaled engineering unit value. The engineering units are generated as BCD and can be the full range of 0 to 9999 (see ANSCLB - Analog Scale 12-Bit Binary to Binary if your raw units are in Binary format).



Note that this IBox only works with unipolar unsigned raw values. It does NOT work with bipolar or sign plus magnitude raw values.

ANSCL Parameters

- Raw (0 to 4095 BCD): specifies the V-memory location of the unipolar unsigned raw 0 to 4095 unscaled value
- High Engineering: specifies the high engineering value when the raw input is 4095
- Low Engineering: specifies the low engineering value when the raw input is 0
- Engineering (BCD): specifies the V-memory location where the scaled engineering BCD value will be placed

ANSCL Example

Parameter		DL205 Range
Raw (0-4095 BCD)	V,P	See DL205 V-memory map - Data Words
High Engineering	K	K0-9999
Low Engineering	K	K0-9999
Engineering (BCD)	V,P	See DL205 V-memory map - Data Words

In the following example, the ANSCL instruction is used to scale a raw value (0 to 4095 BCD) that is in V2000. The engineering scaling range is set 0 to 100 (low engineering value - high engineering value). The scaled value will be placed in V2100 in BCD format.

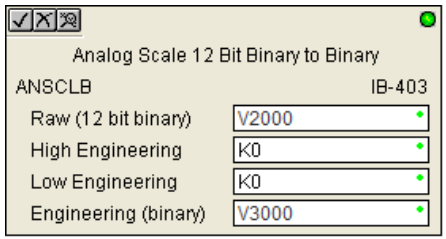
Analog Scale 12-Bit Binary to Binary (ANSCLB) (IB-403)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Analog Scale 12-Bit Binary to Binary scales a 12-bit binary analog value (0 to 4095 decimal) into binary (decimal) engineering units. You specify the engineering unit high value (when raw is 4095), and the engineering low value (when raw is 0), and the output V-memory address where you want to place the scaled engineering unit value. The engineering units are generated as binary and can be the full range of 0 to 65535 (see ANSCL - Analog Scale 12-Bit BCD to BCD if your raw units are in BCD format).

Note that this IBox only works with unipolar unsigned raw values. It does NOT work with bipolar, sign plus magnitude, or signed 2's complement raw values.



ANSCLB Parameters

- Raw (12-bit binary): specifies the V-memory location of the unipolar unsigned raw decimal unscaled value (12-bit binary = 0 to 4095 decimal)
- High Engineering: specifies the high engineering value when the raw input is 4095 decimal
- Low Engineering: specifies the low engineering value when the raw input is 0 decimal
- Engineering (binary): specifies the V-memory location where the scaled engineering decimal value will be placed

ANSCLB Example

Parameter		DL205 Range
Raw (12-bit binary)	V,P	See DL205 V-memory map - Data Words
High Engineering	K	K0-65535
Low Engineering	K	K0-65535
Engineering (binary)	V,P	See DL205 V-memory map - Data Words

In the following example, the ANSCLB instruction is used to scale a raw value (0 to 4095 binary) that is in V2000. The engineering scaling range is set 0 to 1000 (low engineering value - high engineering value). The scaled value will be placed in V2100 in binary format.

Filter Over Time - BCD (FILTER) (IB-422)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Filter Over Time BCD will perform a first-order filter on the Raw Data on a defined time interval. The equation is:

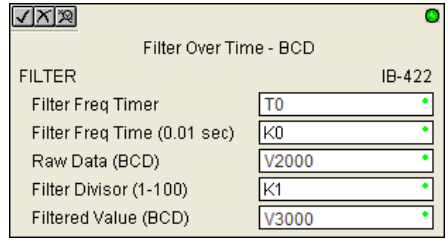
New = Old + [(Raw - Old) / FDC] where,

New: New Filtered Value

Old: Old Filtered Value

FDC: Filter Divisor Constant

Raw: Raw Data



The Filter Divisor Constant is an integer in the range K1 to K100, such that if it equaled K1 then no filtering would be done.

The rate at which the calculation is performed is specified by time in hundredths of a second (0.01 seconds) as the Filter Freq Time parameter. Note that this Timer instruction is embedded in the IBox and must NOT be used anywhere else in your program. Power flow controls whether the calculation is enabled. If it is disabled, the Filter Value is not updated. On the first scan from Program to Run mode, the Filter Value is initialized to 0 to give the calculation a consistent starting point.

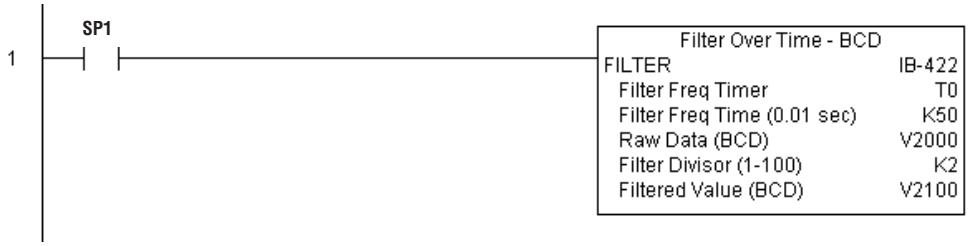
FILTER Parameters

- Filter Frequency Timer: specifies the Timer (T) number which is used by the Filter instruction.
- Filter Frequency Time (0.01sec): specifies the rate at which the calculation is performed.
- Raw Data (BCD): specifies the V-memory location of the raw unfiltered BCD value.
- Filter Divisor (1 to 100): this constant is used to control the filtering effect. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering.
- Filtered Value (BCD): specifies the V-memory location where the filtered BCD value will be placed.

Parameter		DL205 Range
Filter Frequency Timer	T	T0-377
Filter Frequency Time (0.01 sec)	K	K0-9999
Raw Data (BCD)	V	See DL205 V-memory map - Data Words
Filter Divisor (1-100)	K	K1-100
Filtered Value (BCD)	V	See DL205 V-memory map - Data Words

FILTER Example

In the following example, the Filter instruction is used to filter a BCD value that is in V2000. Timer(T0) is set to 0.5 sec, the rate at which the filter calculation will be performed. The filter constant is set to 2. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering. The filtered value will be placed in V2100.



Filter Over Time - Binary (FILTERB) (IB-402)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Filter Over Time in Binary (decimal) will perform a first-order filter on the Raw Data on a defined time interval. The equation is:

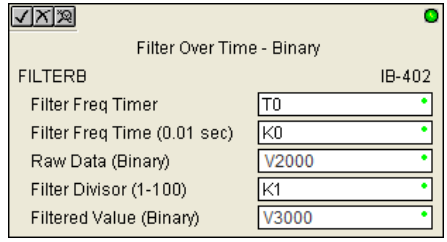
New = Old + [(Raw - Old) / FDC] where

New: New Filtered Value

Old: Old Filtered Value

FDC: Filter Divisor Constant

Raw: Raw Data



The Filter Divisor Constant is an integer in the range K1 to K100, such that if it equaled K1 then no filtering would be done.

The rate at which the calculation is performed is specified by time in hundredths of a second (0.01 seconds) as the Filter Freq Time parameter. Note that this Timer instruction is embedded in the IBox and must NOT be used anywhere else in your program. Power flow controls whether the calculation is enabled. If it is disabled, the Filter Value is not updated. On the first scan from Program to Run mode, the Filter Value is initialized to 0 to give the calculation a consistent starting point.

FILTERB Parameters

- Filter Frequency Timer: specifies the Timer (T) number that is used by the Filter instruction.
- Filter Frequency Time (0.01sec): specifies the rate at which the calculation is performed.
- Raw Data (Binary): specifies the V-memory location of the raw unfiltered binary (decimal) value.
- Filter Divisor (1 to 100): this constant is used to control the filtering effect. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering.
- Filtered Value (Binary): specifies the V-memory location where the filtered binary (decimal) value will be placed.

Parameter		DL205 Range
Filter Frequency Timer	T	T0-377
Filter Frequency Time (0.01 sec)	K	K0-9999
Raw Data (Binary)	V	See DL205 V-memory map - Data Words
Filter Divisor (1-100)	K	K1-100
Filtered Value (Binary)	V	See DL205 V-memory map - Data Words

FILTERB Example

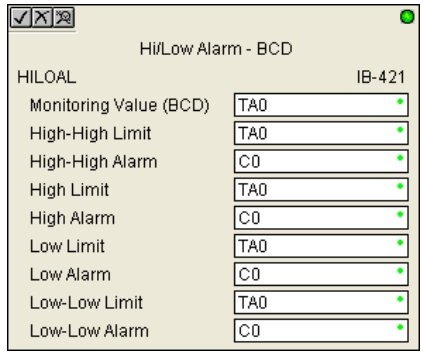
In the following example, the FILTERB instruction is used to filter a binary value that is in V2000. Timer(T1) is set to 0.5 sec, the rate at which the filter calculation will be performed. The filter constant is set to 3. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering. The filtered value will be placed in V2100

Hi/Low Alarm - BCD (HILOAL) (IB-421)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Hi/Low Alarm - BCD monitors a BCD value V-memory location and sets four possible alarm states, High-High, High, Low, and Low-Low whenever the IBox has power flow. You enter the alarm thresholds as constant (K) BCD values (K0-K9999) and/or BCD value V-memory locations.



You must ensure that threshold limits are valid, that is HH >= H > L >= LL. Note that when the High-High or Low-Low alarm condition is true, that the High and Low alarms will also be set, respectively. This means you may use the same threshold limit and same alarm bit for the High-High and the High alarms in case you only need one "High" alarm. Also note that the boundary conditions are inclusive. That is, if the Low boundary is K50, and the Low-Low boundary is K10, and if the Monitoring Value equals 10, then the Low Alarm AND the Low-Low alarm will both be ON. If there is no power flow to the IBox, then all alarm bits will be turned off regardless of the value of the Monitoring Value parameter.

HILOAL Parameters

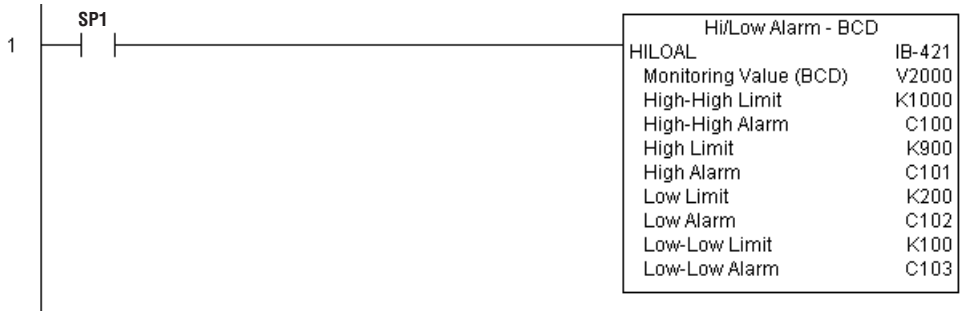
- Monitoring Value (BCD): specifies the V-memory location of the BCD value to be monitored
- High-High Limit: V-memory location or constant specifies the high-high alarm limit
- High-High Alarm: On when the high-high limit is reached
- High Limit: V-memory location or constant specifies the high alarm limit
- High Alarm: On when the high limit is reached
- Low Limit: V-memory location or constant specifies the low alarm limit
- Low Alarm: On when the low limit is reached
- Low-Low Limit: V-memory location or constant specifies the low-low alarm limit
- Low-Low Alarm: On when the low-low limit is reached

Parameter		DL205 Range
Monitoring Value (BCD)	V	See DL205 V-memory map - Data Words
High-High Limit	V, K	K0-9999; or see DL205 V-memory map - Data Words
High-High Alarm	X, Y, C, GX,GY, B	See DL205 V-memory map
High Limit	V, K	K0-9999; or see DL205 V-memory map - Data Words
High Alarm	X, Y, C, GX,GY, B	See DL205 V-memory map
Low Limit	V, K	K0-9999; or see DL205 V-memory map - Data Words
Low Alarm	X, Y, C, GX,GY,B	See DL205 V-memory map
Low-Low Limit	V, K	K0-9999; or see DL205 V-memory map - Data Words
Low-Low Alarm	X, Y, C, GX,GY, B	See DL205 V-memory map

HILOAL Example

In the following example, the HILOAL instruction is used to monitor a BCD value that is in V2000. If the value in V2000 meets/exceeds the High limit of K900, C101 will turn on. If the value continues to increase to meet/exceed the High-High limit, C100 will turn on. Both bits would be on in this case. The High and High-High limits and alarms can be set to the same value if one "High" limit or alarm is desired to be used.

If the value in V2000 meets or falls below the Low limit of K200, C102 will turn on. If the value continues to decrease to meet or fall below the Low-Low limit of K100, C103 will turn on. Both bits would be on in this case. The Low and Low-Low limits and alarms can be set to the same value if one "Low" limit or alarm is desired to be used.



Hi/Low Alarm - Binary (HILOALB) (IB-401)

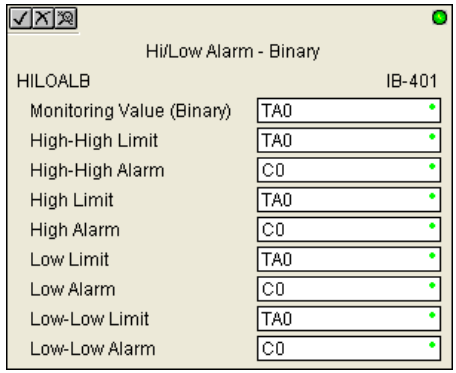
☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Hi/Low Alarm - Binary monitors a binary (decimal) V-memory location and sets four possible alarm states, High-High, High, Low, and Low-Low whenever the IBox has power flow. You enter the alarm thresholds as constant (K) decimal values (K0-K65535) and/or binary (decimal) V-memory locations.

You must ensure that threshold limits are valid, that is HH >= H > L >= LL. Note that when the High-High or Low-Low alarm condition is true, that the High and Low alarms will also be set, respectively. This means you may use the same threshold limit and same alarm bit for the High-High and the High alarms in case you only need one "High" alarm. Also note that the boundary conditions are inclusive. That is, if the Low boundary is K50, and the Low-Low boundary is K10, and if the Monitoring Value equals 10, then the Low Alarm AND the Low-Low alarm will both be ON. If there is no power flow to the IBox, then all alarm bits will be turned off regardless of the value of the Monitoring Value parameter.

HILOALB Parameters



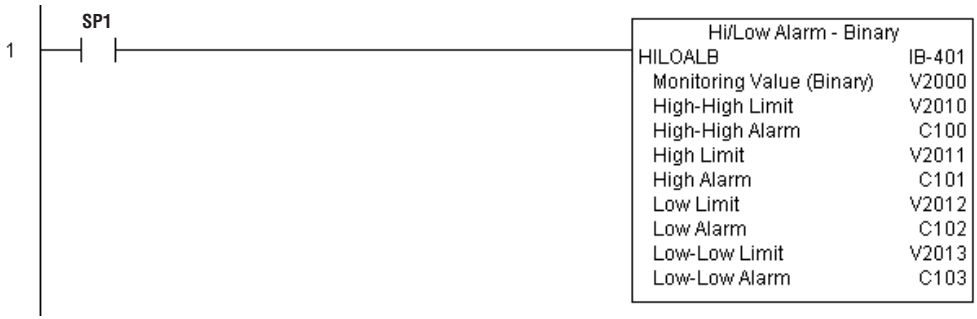
- Monitoring Value (Binary): specifies the V-memory location of the Binary value to be monitored
- High-High Limit: V-memory location or constant specifies the High-High alarm limit
- High-High Alarm: On when the High-High limit is reached
- High Limit: V-memory location or constant specifies the High alarm limit
- High Alarm: On when the High limit is reached
- Low Limit: V-memory location or constant specifies the Low alarm limit
- Low Alarm: On when the Low limit is reached
- Low-Low Limit: V-memory location or constant specifies the Low-Low alarm limit
- Low-Low Alarm: On when the Low-Low limit is reached

Parameter		DL205 Range
Monitoring Value (Binary)	V	See DL205 V-memory map - Data Words
High-High Limit	V, K	K0-65535; or see DL205 V-memory map - Data Words
High-High Alarm	X, Y, C, GX,GY, B	See DL205 V-memory map
High Limit	V, K	K0-65535; or see DL205 V-memory map - Data Words
High Alarm	X, Y, C, GX,GY, B	See DL205 V-memory map
Low Limit	V, K	K0-65535; or see DL205 V-memory map - Data Words
Low Alarm	X, Y, C, GX,GY,B	See DL205 V-memory map
Low-Low Limit	V, K	K0-65535; or see DL205 V-memory map - Data Words
Low-Low Alarm	X, Y, C, GX,GY, B	See DL205 V-memory map

HILOALB Example

In the following example, the HILOALB instruction is used to monitor a binary value that is in V2000. If the value in V2000 meets/exceeds the High limit of the binary value in V2011, C101 will turn on. If the value continues to increase to meet/exceed the High-High limit value in V2010, C100 will turn on. Both bits would be on in this case. The High and High-High limits and alarms can be set to the same V-memory location/value if one "High" limit or alarm is desired to be used.

If the value in V2000 meets or falls below the low limit of the binary value in V2012, C102 will turn on. If the value continues to decrease to meet or fall below the Low-Low limit in V2013, C103 will turn on. Both bits would be on in this case. The Low and Low-Low limits and alarms can be set to the same V-memory location/value if one "Low" limit or alarm is desired to be used.



Off Delay Timer (OFFDTMR) (IB-302)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

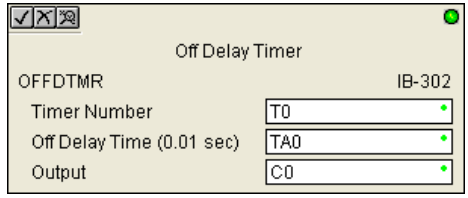
DS5	Used
HPP	N/A

Off Delay Timer will delay the "turning off" of the Output parameter by the specified Off Delay Time (up to 99.99 seconds) based on the power flow into the IBox. Once the IBox receives power, the Output bit will turn on immediately. When the power flow to the IBox turns off, the Output bit WILL REMAIN ON for the specified amount of time (in hundredths of a second). Once the Off Delay Time has expired, the output will turn Off. If the power flow to the IBox comes back on BEFORE the Off Delay Time, then the timer is RESET and the Output will remain On - so you must continuously have NO power flow to the IBox for AT LEAST the specified Off Delay Time before the Output will turn Off.

This IBox utilizes a Timer resource (TMRF), which cannot be used anywhere else in your program.

OFFDTMR Parameters

- Timer Number: specifies the Timer (TMRF) number which is used by the OFFDTMR instruction
- Off Delay Time (0.01sec): specifies how long the Output will remain on once power flow to the Ibox is removed (up to 99.99 seconds).
- Output: specifies the output that will be delayed "turning off" by the Off Delay Time.



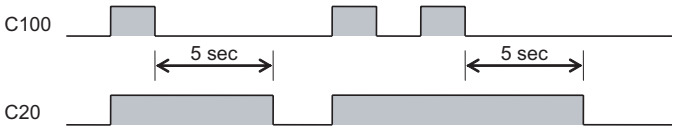
Parameter		DL205 Range
Timer Number	T	T0-377
Off Delay Time	K,V	K0-9999; See DL205 V-memory map - Data Words
Output	X, Y, C, GX,GY, B	See DL205 V-memory map

OFFDTMR Example

In the following example, the OFFDTMR instruction is used to delay the "turning off"of output C20. Timer 2 (T2) is set to 5 seconds, the "off-delay" period.

When C100 turns on, C20 turns on and will remain on while C100 is on. When C100 turns off, C20 will remain on for the specified Off Delay Time (5 secs), and then turn off.

Example timing diagram



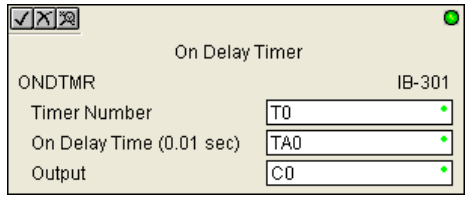
On Delay Timer (ONDTMR) (IB-301)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

On Delay Timer will delay the "turning on" of the Output parameter by the specified amount of time (up to 99.99 seconds) based on the power flow into the IBox. Once the IBox loses power, the Output is turned off immediately. If the power flow turns off BEFORE the On Delay Time, then the timer is RESET and the Output is never turned on, so you must have continuous power flow to the IBox for at least the specified On Delay Time before the Output turns On.

This IBox utilizes a Timer resource (TMRF), which cannot be used anywhere else in your program.



ONDTMR Parameters

- Timer Number: specifies the Timer (TMRF) number which is used by the ONDTMR instruction
- On Delay Time (0.01sec): specifies how long the Output will remain on once power flow to the Ibox is removed (up to 99.99 seconds).
- Output: specifies the output that will be delayed "turning on" by the On Delay Time.

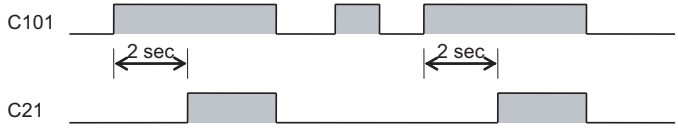
Parameter		DL205 Range
Timer Number	T	T0-377
On Delay Time	K,V	K0-9999; See DL205 V-memory map - Data Words
Output	X, Y, C, GX,GY, B	See DL205 V-memory map

ONDTMR Example

In the following example, the ONDTMR instruction is used to delay the "turning on" of output C21. Timer 1 (T1) is set to 2 seconds, the "on-delay" period.

When C101 turns on, C21 is delayed turning on by 2 seconds. When C101 turns off, C21 turns off immediately.

Example timing diagram



One Shot (ONESHOT) (IB-303)

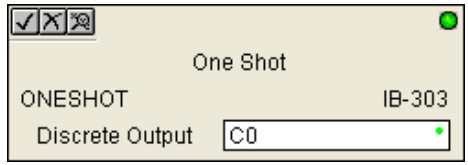
☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

One Shot will turn on the given bit output parameter for one scan on an OFF to ON transition of the power flow into the IBox. This IBox is simply a different name for the PD Coil (Positive Differential).

ONESHOT Parameters

- Discrete Output: specifies the output that will be on for one scan

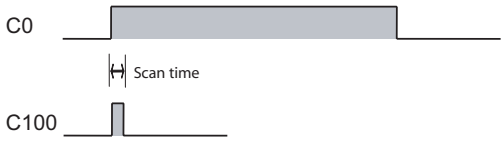


Parameter		DL205 Range
Discrete Output	X, Y, C	See DL205 V-memory map

ONESHOT Example

In the following example, the ONESHOT instruction is used to turn C100 on for one PLC scan after C0 goes from an off to on transition. The input logic must produce an off to on transition to execute the One Shot instruction.

Example Timing Diagram



Push On/Push Off Circuit (PONOFF) (IB-300)

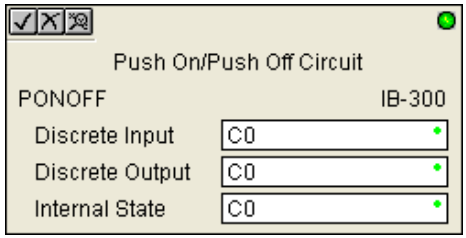
☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Push On/Push Off Circuit toggles an output state whenever its input power flow transitions from off to on. Requires an extra bit parameter for scan-to-scan state information. This extra bit must NOT be used anywhere else in the program. This is also known as a "flip-flop circuit."

PONOFF Parameters

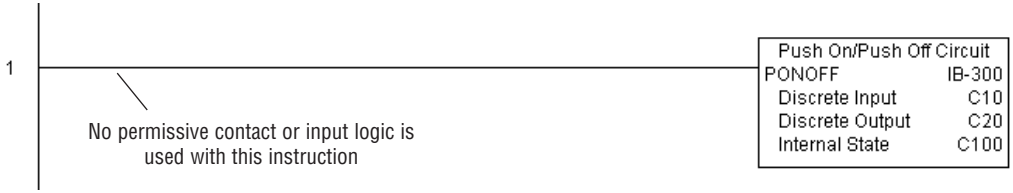
- Discrete Input: specifies the input that will toggle the specified output
- Discrete Output: specifies the output that will be "turned on/off" or toggled
- Internal State: specifies a work bit that is used by the instruction



Parameter		DL205 Range
Discrete Input	X,Y,C,S,T,CT,GX,GY,SP,B,PB	See DL205 V-memory map
Discrete Output	X,Y,C,GX,GY,B	See DL205 V-memory map
Internal State	X, Y, C	See DL205 V-memory map

PONOFF Example

In the following example, the PONOFF instruction is used to control the on and off states of the output C20 with a single input C10. When C10 is pressed once, C20 turns on. When C10 is pressed again, C20 turns off. C100 is an internal bit used by the instruction.



***NOTE**: Neither a permissive nor input logic is required with this instruction.*

Move Single Word (MOVEW) (IB-200)

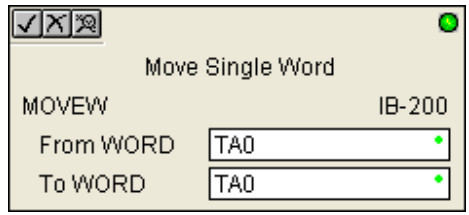
☐ 230
☐ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Move Single Word moves (copies) a word to a memory location directly or indirectly via a pointer, either as a HEX constant, from a memory location, or indirectly through a pointer.

MOVEW Parameters

- From WORD: specifies the word that wi moved to another location
- To WORD: specifies the location to whi where the "From WORD" will be movec



Parameter		DL205 Range
From WORD	V,P,K	K0-FFFF; See DL205 V-memory map - Data Words
To WORD	V,P	See DL205 V-memory map - Data Words

MOVEW Example

In the following example, the MOVEW instruction is used to move 16 bits of data from V2000 to V3000 when C100 turns on.

Move Double Word (MOVED) (IB-201)

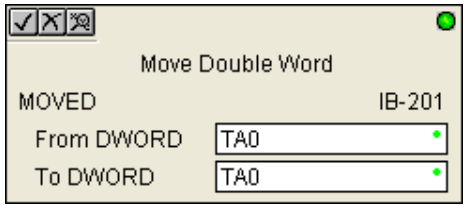
☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Move Double Word moves (copies) a double word to two consecutive memory locations directly or indirectly via a pointer, either as a double HEX constant, from a double memory location, or indirectly through a pointer to a double memory location.

MOVED Parameters

- From DWORD: specifies the double word that will be moved to another location
- To DWORD: specifies the location to which where the "From DWORD" will be moved



Parameter		DL205 Range
From DWORD	V,P,K	K0-FFFFFFFF; See DL205 V-memory map - Data Words
To DWORD	V,P	See DL205 V-memory map - Data Words

MOVED Example

In the following example, the MOVED instruction is used to move 32 bits of data from V2000 and V2001 to V3000 and V3001 when C100 turns on.

BCD to Real with Implied Decimal Point (BCDTOR) (IB-560)

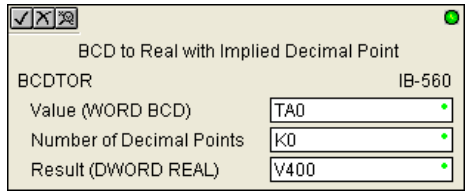
☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

BCD to Real with Implied Decimal Point converts the given 4-digit WORD BCD value to a Real number, with the implied number of decimal points (K0-K4).

For example, BCDTOR K1234 with an implied number of decimal points equal to K1, would yield R123.4

BCDTOR Parameters



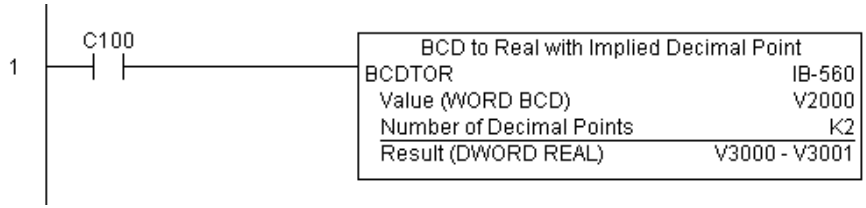
- Value (WORD BCD): specifies the word or constant that will be converted to a Real number
- Number of Decimal Points: specifies the number of implied decimal points in the Result DWORD
- Result (DWORD REAL): specifies the location where the Real number will be placed

Parameter		DL205 Range
Value (WORD BCD)	V,P,K	K0-9999; See DL205 V-memory map - Data Words
Number of Decimal Points	K	K0-4
Result (DWORD REAL)	V	See DL205 V-memory map - Data Words

BCDTOR Example

In the following example, the BCDTOR instruction is used to convert the 16-bit data in V2000 from a 4-digit BCD data format to a 32-bit REAL (floating point) data format and store into V3000 and V3001 when C100 turns on.

K2 in the Number of Decimal Points implies the data will have two digits to the right of the decimal point.



Double BCD to Real with Implied Decimal Point (BCDTORD) (IB-562)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

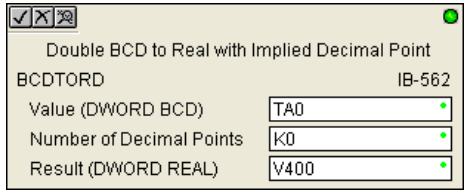
DS5	Used
HPP	N/A

Double BCD to Real with Implied Decimal Point converts the given 8-digit DWORD BCD value to a Real number, given an implied number of decimal points (K0-K8).

For example, BCDTORD K12345678 with an implied number of decimal points equal to K5, would yield R123.45678

BCDTORD Parameters

- Value (DWORD BCD): specifies the Dword or constant that will be converted to a Real number
- Number of Decimal Points: specifies the number of implied decimal points in the Result DWORD
- Result (DWORD REAL): specifies the location where the Real number will be placed

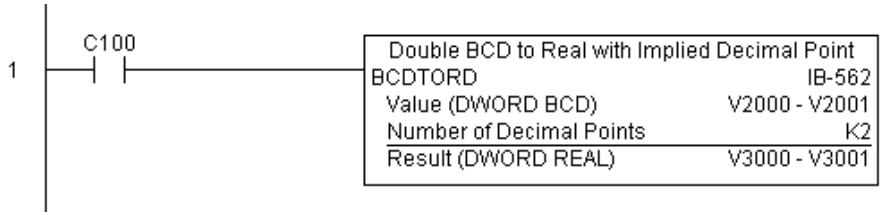


Parameter		DL205 Range
Value (DWORD BCD)	V,P,K	K0-99999999; See DL205 V-memory map - Data Words
Number of Decimal Points	K	K0-8
Result (DWORD REAL)	V	See DL205 V-memory map - Data Words

BCDTORD Example

In the following example, the BCDTORD instruction is used to convert the 32-bit data in V2000 from an 8-digit BCD data format to a 32-bit REAL (floating point) data format and store into V3000 and V3001 when C100 turns on.

K2 in the Number of Decimal Points implies the data will have two digits to the right of the decimal point.

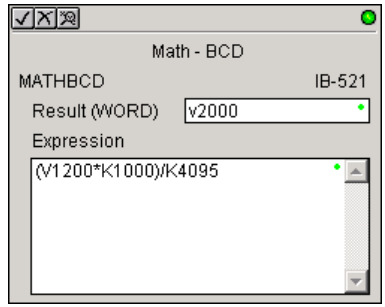


Math - BCD (MATHBCD) (IB-521)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Math - BCD Format lets you enter complex mathematical expressions like you would in Visual Basic, Excel, or C++ to do complex calculations, nesting parentheses up to 4 levels deep. In addition to + - * /, you can do Modulo (% aka Remainder), Bit-wise And (&) Or (|) Xor (^), and some BCD functions - Convert to BCD (BCD), Convert to Binary (BIN), BCD Complement (BCDCPL), Convert from Gray Code (GRAY), Invert Bits (INV), and BCD/HEX to Seven Segment Display (SEG).



Example: ((V2000 + V2001) / (V2003 - K100)) * GRAY(V3000 & K001F)

Every V-memory reference MUST be to a single-word BCD formatted value. Intermediate results can go up to 32-bit values, but as long as the final result fits in a 16-bit BCD word, the calculation is valid. Typical example of this is scaling using multiply then divide, (V2000 * K1000) / K4095. The multiply term most likely will exceed 9999 but fits within 32 bits. The divide operation will divide 4095 into the 32-bit accumulator, yielding a result that will always fit in 16 bits.

You can reference binary V-memory values by using the BCD conversion function on a V-memory location but NOT an expression. That is BCD(V2000) is okay and will convert V2000 from Binary to BCD, but BCD(V2000 + V3000) will add V2000 as BCD, to V3000 as BCD, then interpret the result as Binary and convert it to BCD - NOT GOOD.

Also, the final result is a 16-bit BCD number and so you could do BIN around the entire operation to store the result as Binary.

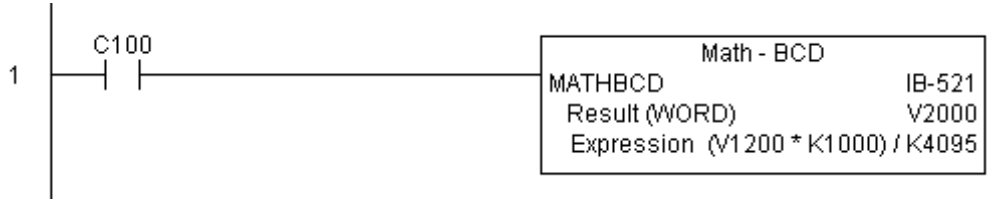
MATHBCD Parameters

- Result (WORD): specifies the location where the BCD result of the mathematical expression will be placed (result must fit into 16-bit single V-memory location).
- Expression: specifies the mathematical expression to be executed and the result is stored in specified Result (WORD). Each V-memory location used in the expression must be in BCD format.

Parameter		DL205 Range
WORD Result	V	See DL205 V-memory map - Data Words
Expression		Text

MATHBCD Example

In the following example, the MATHBCD instruction is used to calculate the math expression which multiplies the BCD value in V1200 by 1000, then divides by 4095 and loads the resulting value in V2000 when C100 turns on.

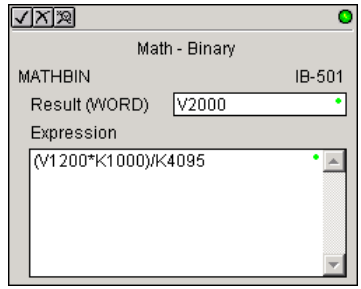


Math - Binary (MATHBIN) (IB-501)

☐ 230
☐ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Math - Binary Format lets you enter complex mathematical expressions like you would in Visual Basic, Excel, or C++ to do complex calculations, nesting parentheses up to 4 levels deep. In addition to + - * /, you can do Modulo (% aka Remainder), Shift Right (>>) and Shift Left (<<), Bit-wise And (&) Or (|) Xor (^), and some binary functions - Convert to BCD (BCD), Convert to Binary (BIN), Decode Bits (DECO), Encode Bits (ENCO), Invert Bits (INV), HEX to Seven Segment Display (SEG), and Sum Bits (SUM).



Example: ((V2000 + V2001) / (V2003 - K10)) * SUM(V3000 & K001F)

Every V-memory reference MUST be to a single-word binary formatted value. Intermediate results can go up to 32-bit values, but as long as the final result fits in a 16-bit binary word, the calculation is valid. Typical example of this is scaling using multiply then divide, (V2000 * K1000) / K4095. The multiply term most likely will exceed 65535 but fits within 32 bits. The divide operation will divide 4095 into the 32-bit accumulator, yielding a result that will always fit in 16 bits.

You can reference BCD V-memory values by using the BIN conversion function on a V-memory location but NOT an expression. That is, BIN(V2000) is okay and will convert V2000 from BCD to Binary, but BIN(V2000 + V3000) will add V2000 as Binary, to V3000 as Binary, then interpret the result as BCD and convert it to Binary - NOT GOOD.

Also, the final result is a 16-bit binary number and so you could do BCD around the entire operation to store the result as BCD.

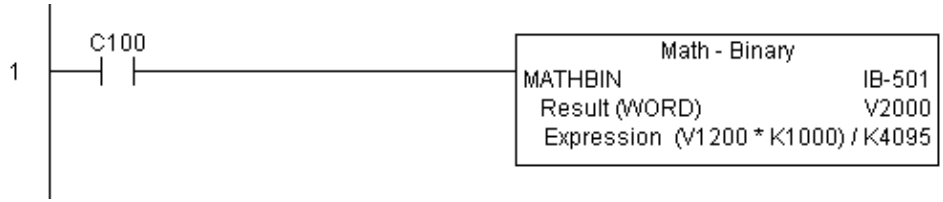
MATHBIN Parameters

- Result (WORD): specifies the location where the binary result of the mathematical expression will be placed (result must fit into 16-bit single V-memory location).
- Expression: specifies the mathematical expression to be executed and the result is stored in specified Result (WORD). Each V-memory location used in the expression must be in binary format.

Parameter		DL205 Range
WORD Result	V	See DL205 V-memory map - Data Words
Expression		Text

MATHBIN Example

In the following example, the MATHBIN instruction is used to calculate the math expression which multiplies the Binary value in V1200 by 1000 then divides by 4095 and loads the resulting value in V2000 when C100 turns on.

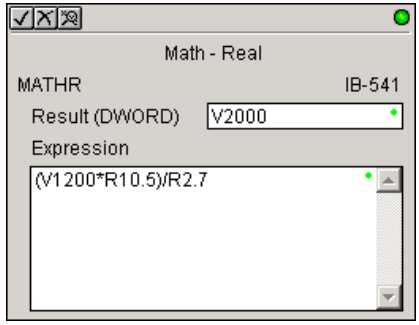


Math - Real (MATHR) (IB-541)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Math - Real Format lets you enter complex mathematical expressions like you would in Visual Basic, Excel, or C++ to do complex calculations, nesting parentheses up to 4 levels deep. In addition to + - * /, you can do Bit-wise And (&) Or (|) and Xor (^). The D2-260 and D2-262 also support several Real functions - Arc Cosine (ACOSR), Arc Sine (ASINR), Arc Tangent (ATANR), Cosine (COSR), Convert Radians to Degrees (DEGR), Invert Bits (INV), Convert Degrees to Radians (RADR), HEX to Seven Segment Display (SEG), Sine (SINR), Square Root (SQRTR), and Tangent (TANR).



Example: ((V2000 + V2002) / (V2004 - R2.5)) * SINR(RADR(V3000 / R10.0))

Every V-memory reference MUST be able to fit into a double-word Real formatted value.

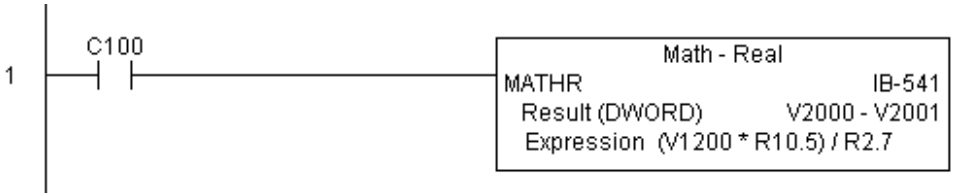
MATHR Parameters

- Result (DWORD): specifies the location where the Real result of the mathematical expression will be placed (result must fit into a double-word Real formatted location).
- Expression: specifies the mathematical expression to be executed and the result is stored in specified Result (DWORD) location. Each V-memory location used in the expression must be in Real format.

Parameter		DL205 Range
DWORD Result	V	See DL205 V-memory map - Data Words
Expression		Text

MATHR Example

In the following example, the MATHR instruction is used to calculate the math expression which multiplies the REAL (floating point) value in V1200 by 10.5 then divides by 2.7 and loads the resulting 32-bit value in V2000 and V2001 when C100 turns on.



Real to BCD with Implied Decimal Point and Rounding (RTOBCD) (IB-561)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

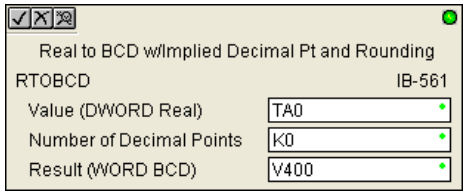
Real to BCD with Implied Decimal Point and Rounding converts the absolute value of the given Real number to a 4-digit BCD number, compensating for an implied number of decimal points (K0-K4) and performs rounding.

For example, RTOBCD R56.74 with an implied number of decimal points equal to K1, would yield 567 BCD. If the implied number of decimal points was 0, then the function would yield 57 BCD (note that it rounded up).

If the Real number is negative, the Result will equal its positive, absolute value.

RTOBCD Parameters

- Value (DWORD Real): specifies the Real Dword location or number that will be converted and rounded to a BCD number with decimal points
- Number of Decimal Points: specifies the number of implied decimal points in the Result WORD
- Result (WORD BCD): specifies the location where the rounded/implied decimal points BCD value will be placed

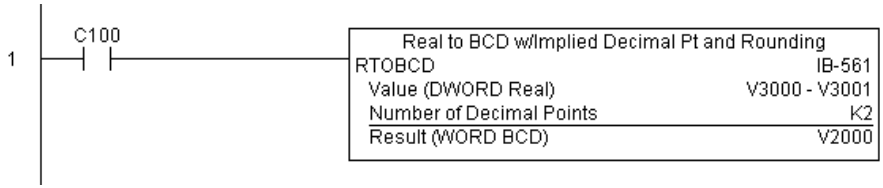


Parameter		DL205 Range
Value (DWORD Real)	V,P,R	R ; See DL205 V-memory map - Data Words
Number of Decimal Points	K	K0-4
Result (WORD BCD)	V	See DL205 V-memory map - Data Words

RTOBCD Example

In the following example, the RTOBCD instruction is used to convert the 32-bit REAL (floating point) data format in V3000 and V3001 to the 4-digit BCD data format and store in V2000 when C100 turns on.

K2 in the Number of Decimal Points implies the data will have two implied decimal points.



Real to Double BCD with Implied Decimal Point and Rounding (RTOBCDD) (IB-563)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

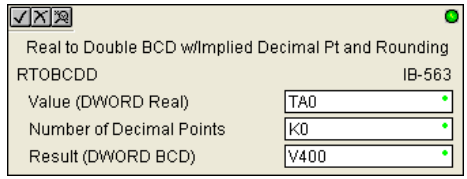
Real to Double BCD with Implied Decimal Point and Rounding converts the absolute value of the given Real number to an 8-digit DWORD BCD number, compensating for an implied number of decimal points (K0-K8) and performs rounding.

For example, RTOBCDD R38156.74 with an implied number of decimal points equal to K1, would yield 381567 BCD. If the implied number of decimal points was 0, then the function would yield 38157 BCD (note that it rounded up).

DS5	Used
HPP	N/A

If the Real number is negative, the Result will equal its positive, absolute value.

RTOBCDD Parameters



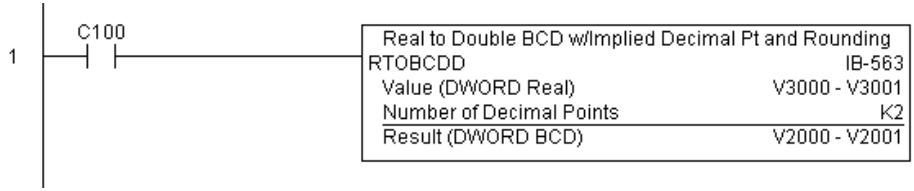
- Value (DWORD Real): specifies the Dword Real number that will be converted and rounded to a BCD number with decimal points
- Number of Decimal Points: specifies the number of implied decimal points in the Result DWORD
- Result (DWORD BCD): specifies the location where the rounded/implied decimal points DWORD BCD value will be placed

Parameter		DL205 Range
Value (DWORD Real)	V,P,R	R ; See DL205 V-memory map - Data Words
Number of Decimal Points	K	K0-8
Result (DWORD BCD)	V	See DL205 V-memory map - Data Words

RTOBCDD Example

In the following example, the RTOBCDD instruction is used to convert the 32-bit REAL (floating point) data format in V3000 and V3001 to the 8-digit BCD data format and store in V2000 and V2001 when C100 turns on.

K2 in the Number of Decimal Points implies the data will have two implied decimal points.



Square BCD (SQUARE) (IB-523)

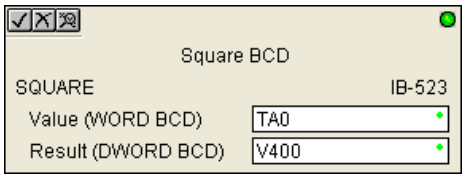
☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Square BCD squares the given 4-digit WORD BCD number and writes it as an 8-digit DWORD BCD result.

SQUARE Parameters

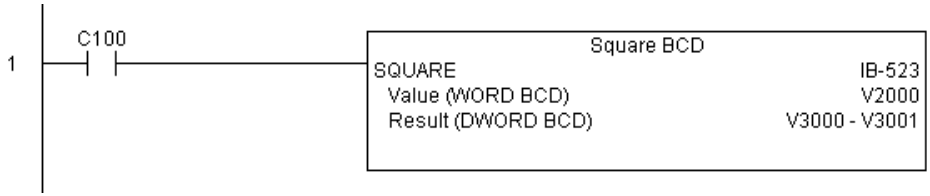
- Value (WORD BCD): specifies the BCD Word or constant that will be squared
- Result (DWORD BCD): specifies the location where the squared DWORD BCD value will be placed



Parameter		DL205 Range
Value (WORD BCD)	V,P,K	K0-9999 ; See DL205 V-memory map - Data Words
Result (DWORD BCD)	V	See DL205 V-memory map - Data Words

SQUARE Example

In the following example, the SQUARE instruction is used to square the 4-digit BCD value in V2000 and store the 8-digit double word BCD result in V3000 and V3001 when C100 turns on.



Square Binary (SQUAREB) (IB-503)

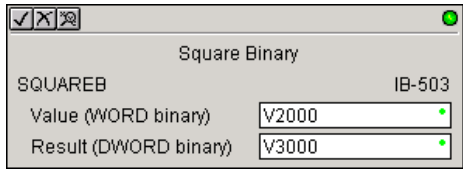
Square Binary squares the given 16-bit WORD Binary number and writes it as a 32-bit DWORD Binary result.

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

SQUAREB Parameters

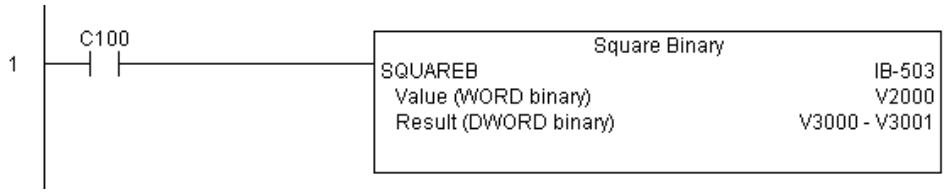
- Value (WORD Binary): specifies the binary Word or constant that will be squared
- Result (DWORD Binary): specifies the location where the squared DWORD binary value will be placed



Parameter		DL205 Range
Value (WORD Binary)	V,P,K	K0-65535; See DL205 V-memory map - Data Words
Result (DWORD Binary)	V	See DL205 V-memory map - Data Words

SQUAREB Example

In the following example, the SQUAREB instruction is used to square the single-word Binary value in V2000 and store the 8-digit double-word Binary result in V3000 and V3001 when C100 turns on.



Square Real (SQUARER) (IB-543)

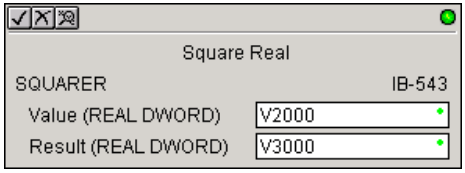
Square Real squares the given REAL DWORD number and writes it to a REAL DWORD result.

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

SQUARER Parameters

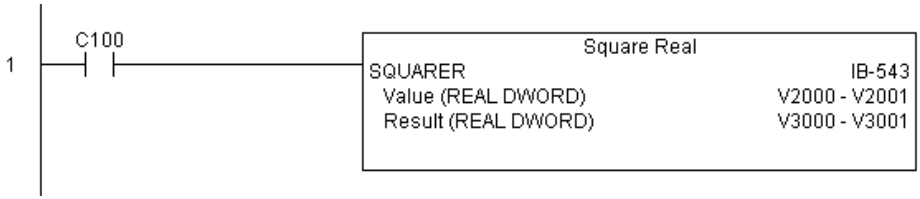
- Value (REAL DWORD): specifies the Real DWORD location or number that will be squared
- Result (REAL DWORD): specifies the location where the squared Real DWORD value will be placed



Parameter		DL205 Range
Value (REAL DWORD)	V,P,R	R ; See DL205 V-memory map - Data Words
Result (REAL DWORD)	V	See DL205 V-memory map - Data Words

SQUARER Example

In the following example, the SQUARER instruction is used to square the 32-bit floating point REAL value in V2000 and V2001 and store the REAL value result in V3000 and V3001 when C100 turns on.



Sum BCD Numbers (SUMBCD) (IB-522)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

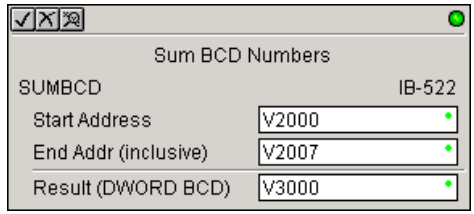
Sum BCD Numbers sums up a list of consecutive 4-digit WORD BCD numbers into an 8-digit DWORD BCD result.

You specify the group's starting and ending V-memory addresses (inclusive). When enabled, this instruction will add all the numbers in the group (so you may want to place a differential contact driving the enable).

SUMBCD could be used as the first part of calculating an average.

DS5	Used
HPP	N/A

SUMBCD Parameters

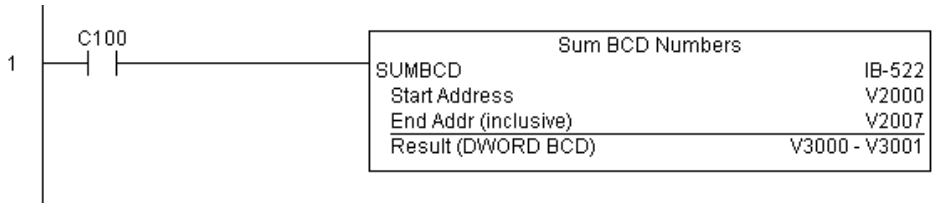


- Start Address: specifies the starting address of a block of V-memory location values to be added together (BCD)
- End Addr (inclusive): specifies the ending address of a block of V-memory location values to be added together (BCD)
- Result (DWORD BCD): specifies the location where the sum of the block of V-memory BCD values will be placed

Parameter		DL205 Range
Start Address	V	See DL205 V-memory map - Data Words
End Address (inclusive)	V	See DL205 V-memory map - Data Words
Result (DWORD BCD)	V	See DL205 V-memory map - Data Words

SUMBCD Example

In the following example, the SUMBCD instruction is used to total the sum of all BCD values in words V2000 thru V2007 and store the resulting 8-digit double word BCD value in V3000 and V3001 when C100 turns on.



Sum Binary Numbers (SUMBIN) (IB-502)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

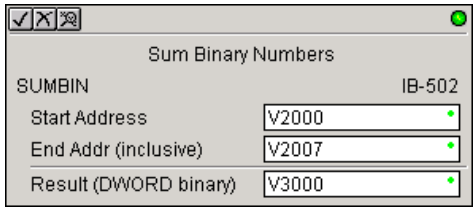
Sum Binary Numbers sums up a list of consecutive 16-bit WORD Binary numbers into a 32-bit DWORD binary result.

You specify the group's starting and ending V-memory addresses (inclusive). When enabled, this instruction will add all the numbers in the group (so you may want to place a differential contact driving the enable).

SUMBIN could be used as the first part of calculating an average.

DS5	Used
HPP	N/A

SUMBIN Parameters

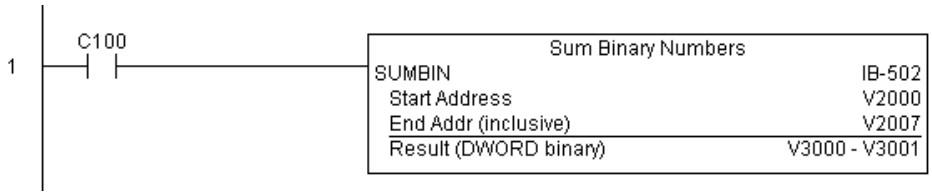


- Start Address: specifies the starting address of a block of V-memory location values to be added together (Binary)
- End Addr (inclusive): specifies the ending address of a block of V-memory location values to be added together (Binary)
- Result (DWORD Binary): specifies the location where the sum of the block of V-memory binary values will be placed

Parameter		DL205 Range
Start Address	V	See DL205 V-memory map - Data Words
End Address (inclusive)	V	See DL205 V-memory map - Data Words
Result (DWORD Binary)	V	See DL205 V-memory map - Data Words

SUMBIN Example

In the following example, the SUMBIN instruction is used to total the sum of all Binary values in words V2000 thru V2007 and store the resulting 8-digit double word Binary value in V3000 and V3001 when C100 turns on.



Sum Real Numbers (SUMR) (IB-542)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Sum Real Numbers sums up a list of consecutive REAL DWORD numbers into a REAL DWORD result.

You specify the group's starting and ending V-memory addresses (inclusive).

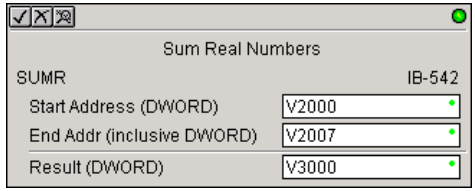
Remember that Real numbers are DWORDs and occupy 2 words of V-memory each, so the number of Real values summed up is equal to half the number of memory locations. Note that the End Address can be EITHER word of the 2 word ending address, for example, if you wanted to add the 4 Real numbers stored in V2000 thru V2007 (V2000, V2002, V2004, and V2006), you can specify V2006 OR V2007 for the ending address and you will get the same result.

When enabled, this instruction will add all the numbers in the group (so you may want to place a differential contact driving the enable).

SUMR could be used as the first part of calculating an average.

SUMR Parameters

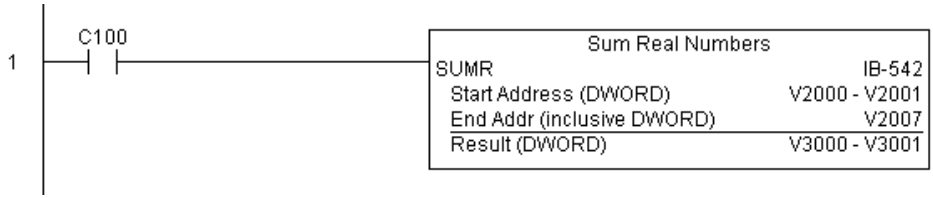
- Start Address (DWORD): specifies the starting address of a block of V-memory location values to be added together (Real)
- End Addr (inclusive) (DWORD): specifies the ending address of a block of V-memory location values to be added together (Real)
- Result (DWORD): specifies the location where the sum of the block of V-memory Real values will be placed



Parameter		DL205 Range
Start Address (DWORD)	V	See DL205 V-memory map - Data Words
End Address (inclusive DWORD)	V	See DL205 V-memory map - Data Words
Result (DWORD)	V	See DL205 V-memory map - Data Words

SUMR Example

In the following example, the SUMR instruction is used to total the sum of all floating point REAL number values in words V2000 thru V2007 and store the resulting 32-bit floating point REAL number value in V3000 and V3001 when C100 turns on.



ECOM100 Configuration (ECOM100) (IB-710)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 Configuration defines all the common information for one specific ECOM100 module which is used by the other ECOM100 IBoxes; for example, ECRX - ECOM100 Network Read, ECEMAIL - ECOM100 Send EMail, ECIPSUP - ECOM100 IP Setup, etc.

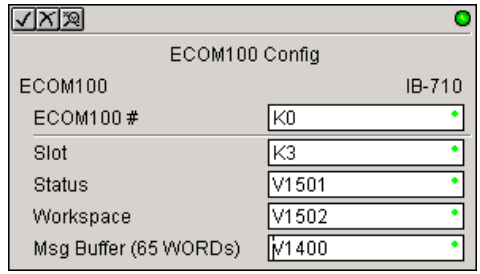
You MUST have the ECOM100 Configuration IBox at the top of your ladder/stage program with any other configuration IBoxes. The Message Buffer parameter specifies the starting address of a 65 WORD buffer. This is 101 Octal addresses (e.g., V1400 thru V1500).

If you have more than one ECOM100 in your PLC, you must have a different ECOM100 Configuration IBox for EACH ECOM100 module in your system that utilizes any ECOM IBox instructions.

The Workspace and Status parameters and the entire Message Buffer are internal, private registers used by the ECOM100 Configuration IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

In order for MOST ECOM100 IBoxes to function, you must turn ON dip switch 7 on the ECOM100 circuit board. You can keep dip switch 7 off if you are ONLY using ECOM100 Network Read and Write IBoxes (ECRX, ECWX).

ECOM100 Parameters



- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Slot: specifies which PLC slot is occupied by the ECOM100 module.
- Status: specifies a V-memory location that will be used by the instruction.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Msg Buffer: specifies the starting address of a 65-word buffer that will be used by the module for configuration.

Parameter		DL205 Range
ECOM100#	K	K0-255
Slot	K	K0-7
Status	V	See DL205 V-memory map - Data Words
Workspace	V	See DL205 V-memory map - Data Words
Msg Buffer (65 words used)	V	See DL205 V-memory map - Data Words

ECOM100 Example

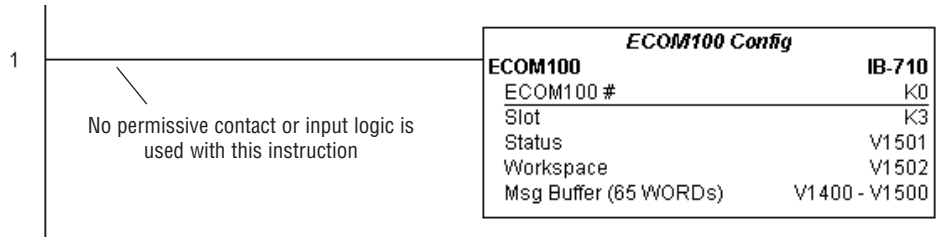
The ECOM100 Config IBox coordinates all of the interaction with other ECOM100-based IBoxes (ECxxxx). You must have an ECOM100 Config IBox for each ECOM100 module in your system. Configuration IBoxes must be at the top of your program and must execute every scan.

This IBox defines ECOM100# K0 to be in slot 3. Any ECOM100 IBoxes that need to reference this specific module (such as ECEMAIL, ECRX, ...) would enter K0 for their ECOM100# parameter.

The Status register is for reporting any completion or error information to other ECOM100 IBoxes. This V-memory register must not be used anywhere else in the entire program.

The Workspace register is used to maintain state information about the ECOM100, along with proper sharing and interlocking with the other ECOM100 IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.

The Message Buffer of 65 words (130 bytes) is a common pool of memory that is used by other ECOM100 IBoxes (such as ECEMAIL). This way, you can have a bunch of ECEMAIL IBoxes, but only need 1 common buffer for generating and sending each EMail. These V-memory registers must not be used anywhere else in your entire program.



NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

ECOM100 Disable DHCP (ECDHCPD) (IB-736)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 Disable DHCP will set up the ECOM100 to use its internal TCP/IP settings on a leading edge transition to the IBox. To configure the ECOM100's TCP/IP settings manually, use the NetEdit3 utility, or you can do it programmatically from your PLC program using the ECOM100 IP Setup (ECIPSUP), or the individual ECOM100 IBoxes: ECOM Write IP Address (ECWRIP), ECOM Write Gateway Address (ECWRGWA), and ECOM100 Write Subnet Mask (ECWRSNM).

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

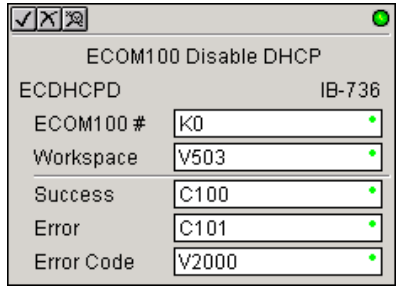
Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The "Disable DHCP" setting is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is HIGHLY RECOMMENDED that you only execute this IBox ONCE, on the second scan. Since it requires a LEADING edge to execute, use a NORMALLY CLOSED SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECDHCPD Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Error Code: specifies the location where the Error Code will be written



Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map - Data Words

ECDHCPD Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: On the 2nd scan, disable DHCP in the ECOM100. DHCP is the same protocol used by PCs for using a DHCP Server to automatically assign the ECOM100's IP Address, Gateway Address, and Subnet Mask. Typically disabling DHCP is done by assigning a hard-coded IP Address either in NetEdit or using one of the ECOM100 IP Setup IBoxes, but this IBox allows you to disable DHCP in the ECOM100 using your ladder program. The ECDHCPD is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to disable DHCP will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON. If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

ECOM100 Enable DHCP (ECDHCPE) (IB-735)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 Enable DHCP will tell the ECOM100 to obtain its TCP/IP setup from a DHCP Server on a leading edge transition to the IBox.

The IBox will be successful once the ECOM100 has received its TCP/IP settings from the DHCP server. Since it is possible for the DHCP server to be unavailable, a Timeout parameter is provided so that the IBox can complete, but with an Error (Error Code = 1004 decimal).

See also the ECOM100 IP Setup (ECIPSUP) IBox 717 to directly set up ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.

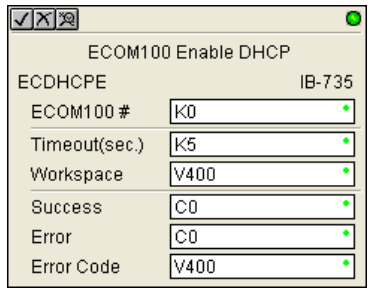
The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The "Enable DHCP" setting is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is HIGHLY RECOMMENDED that you only execute this IBox ONCE, on the second scan. Since it requires a LEADING edge to execute, use a NORMALLY CLOSED SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECDHCPE Parameters



- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Timeout(sec): specifies a timeout period so that the instruction may have time to complete.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Error Code: specifies the location where the Error Code will be written.

Parameter		DL205 Range
ECOM100#	K	K0-255
Timeout (sec)	K	K5-127
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map - Data Words

ECDHCPE Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module.V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: On the second scan, enable DHCP in the ECOM100. DHCP is the same protocol used by PCs for using a DHCP Server to automatically assign the ECOM100's IP Address, Gateway Address, and Subnet Mask. Typically this is done using NetEdit, but this IBox allows you to enable DHCP in the ECOM100 using your ladder program. The ECDHCPE is leading edge triggered, not power-flow driven (similar to a counter input leg). The commands to enable DHCP will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON. The ECDHCPE does more than just set the bit to enable DHCP in the ECOM100, it polls the ECOM100 once every second to see if the ECOM100 has found a DHCP server and has a valid IP Address. Therefore, a timeout parameter is needed in case the ECOM100 cannot find a DHCP server. If a timeout does occur, the Error bit will turn on and the error code will be 1005 decimal. The Success bit will turn on only if the ECOM100 finds a DHCP Server and is assigned a valid IP Address. If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

ECOM100 Query DHCP Setting (ECDHCPQ) (IB-734)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

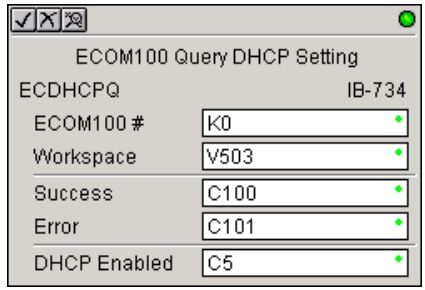
ECOM100 Query DHCP Setting will determine if DHCP is enabled in the ECOM100 on a leading edge transition to the IBox. The DHCP Enabled bit parameter will be ON if DHCP is enabled, OFF if disabled.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECDHCPQ Parameters

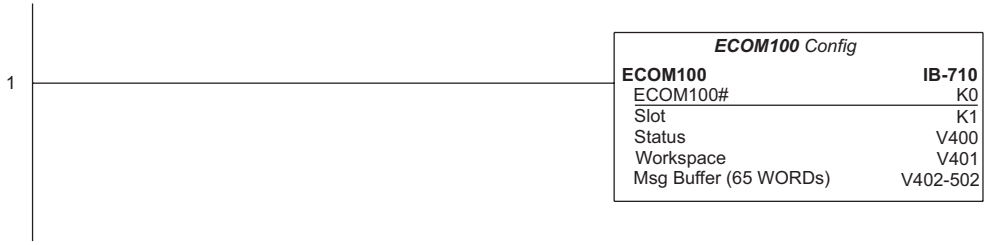


- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the instruction is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- DHCP Enabled: specifies a bit that will turn on if the ECOM100's DHCP is enabled or remain off if disabled - after instruction query, be sure to check the state of the Success/Error bit state along with DHCP Enabled bit state to confirm a successful module query.

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
DHCP Enabled	X,Y,C,GX,GY,B	See DL205 V-memory map

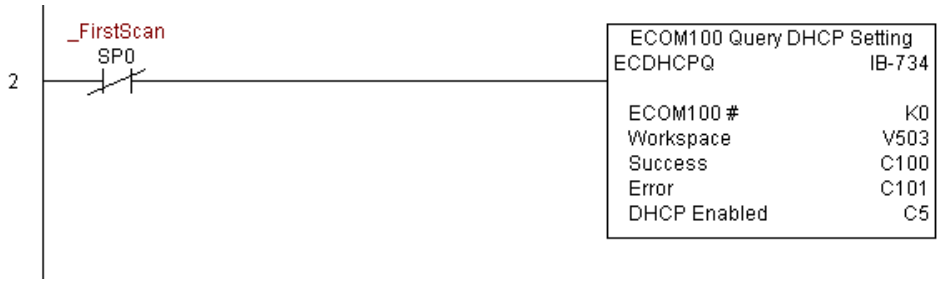
ECDHCPQ Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module.V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

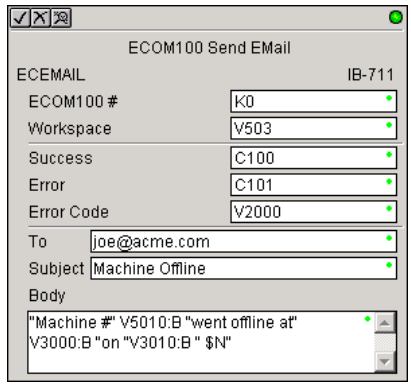
Rung 2: On the second scan, read whether DHCP is enabled or disabled in the ECOM100 and store it in C5. DHCP is the same protocol used by PCs for using a DHCP Server to automatically assign the ECOM100's IP Address, Gateway Address, and Subnet Mask. The ECDHCPQ is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read (Query) whether DHCP is enabled or not will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON. If successful, turn on C100. If there is a failure, turn on C101.



ECOM100 Send E-mail (ECEMAIL) (IB-711)

☒ 230
☒ 240
☑ **250-1**
☑ **260**
☑ **262**

ECOM100 Send EMail, on a leading edge transition, will behave as an EMail client and send an SMTP request to your SMTP Server to send the EMail message to the EMail addresses in the To: field and also to those listed in the Cc: list hard coded in the ECOM100. It will send the SMTP request based on the specified ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) at the top of your program.



DS5	Used
HPP	N/A

The Body: field supports what the PRINT and VPRINT instructions support for text and embedded variables, allowing you to embed real-time data in your EMail (e.g., "V2000 = " V2000:B).

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the request is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), an SMPT protocol error (between 100 and 999), or a PLC logic error (greater than 1000).

Since the ECOM100 is only an EMail Client and requires access to an SMTP Server, you MUST have the SMTP parameters configured properly in the ECOM100 via the ECOM100's Home Page and/or the EMail Setup instruction (ECEMSUP). To get to the ECOM100's Home Page, use your favorite Internet browser and browse to the ECOM100's IP Address, e.g., http://192.168.12.86

You are limited to approximately 100 characters of message data for the entire instruction, including the To: Subject: and Body: fields. To save space, the ECOM100 supports a hard coded list of EMail addresses for the Carbon Copy field (cc:) so that you can configure those IN the ECOM100, and keep the To: field small (or even empty), to leave more room for the Subject: and Body: fields.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECEMAIL Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Error Code: specifies the location where the Error Code will be written
- To: specifies an E-mail address that the message will be sent to
- Subject: subject of the e-mail message
- Body: supports what the PRINT and VPRINT instructions support for text and embedded variables, allowing you to embed real-time data in the EMail message

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map
To:		Text
Subject:		Text
Body:		See PRINT and VPRINT instructions

ECEMAIL Decimal Status Codes

This list of status codes is based on the list in the *ECOM100 Mock Slave Address 89 Command Specification*.

ECOM100 Status codes can be classified into four different areas based on its **decimal** value.

ECOM100 Status Code Areas	
Error	**Description**
0-1	Normal Status - no error
2-99	Internal ECOM100 errors
100-999	Standard TCP/IP protocol errors (SMTP, HTTP, etc)
1000+	IBox ladder logic assigned errors (SP Slot Error, etc)

For the ECOM100 Send EMail IBOX, the status codes below are specific to this IBox.

Normal Status 0-1

ECOMM100 Send EMAIL IBOX Status Codes	
Error	**Description**
0-1	Success - ECEMAIL completed successfully.
1	Busy - ECEMAIL IBOX logic sets the Error register to this value when the ECEMAIL starts a new request.

Internal ECOM100 Errors (2-99)

Internal ECOM 100 Errors (2-99)	
Error	**Description**
10-19	Timeout Errors - last digit shows where in ECOM100's SMTP state logic the timeout occurred; *regardless of the last digit*, the SMTP conversation with the SMTP Server timed out.
	SMTP Internal Errors (20-29)
20	TCP Write Error
21	No Sendee
22	Invalid State
23	Invalid Data
24	Invalid SMTP Configuration
25	Memory Allocation Error

ECEMAIL IBox Ladder Logic Assigned Errors (1000+)

ECEMAIL IBox Ladder Logic Assigned Errors (1000+)	
Error	**Description**
101	SP Slot Error - the SP error bit for the ECOM100's slot turned on. Possibly using RX or WX instructions on the ECOM100 and walking on the ECEMAIL execution. Use should use ECRX and ECX IBoxes,

ECEMAIL Decimal Status Codes

SMTP Protocol Errors - SMTP (100-999)

SMTP Protocol Errors - SMTP (100-999)	
Error	**Description**
1xx	Informational replies
2xx	Success replies
200	(Non-standard success response)
211	System status or system help reply
214	Help message
220	<domain> Service ready. Ready to start TLS
221	<domain> Service closing transmission channel
250	Ok, queuing for node <node> started. Requested mail action okay, completed
251	Ok, no messages waiting for node <node>. User not local; will forward to <forward-path>
252	OK, pending messages for node <node> started. Cannot VRFY user (e.g., info is not local), but will take message for this user and attempt delivery.
253	OK, messages pending messages for node <node> started
3xx`	(re) direction replies
354	Start mail input; end with <CRLF>.<CRLF>
355	Octet-offset is the transaction offset.
4xx	client/request error replies
421	<domain> Service not available, closing transmission channel
432	A password transition is needed
450	Requested mail action not taken: mailbox unavailable. ATRN request refused.
451	Requested action aborted; local error in processing. Unable to process ATRN request now.
452	Requested action not taken: insufficient system storage
453	You have no mail
454	TLS is not available due to temporary reason. Encryption required for requested authentication mechanism.
458	Unable to queue messages for node <node>
459	Node <node> not allowed: <reason>
5xx	Server/process error replies
500	Syntax error, command unrecognized. Syntax error.
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
521	<domain> does not accept mail
530	Access denied. Must issue a STARTTLS command first. Encryption required for requested authentication mechanism.

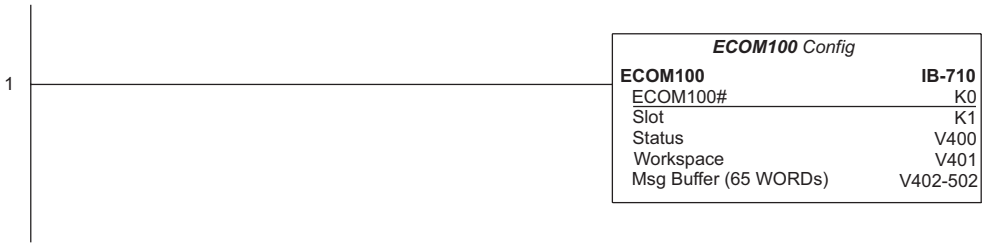
Continued next page

SMTP Protocol Errors - SMTP (100-999)	
Error	**Description**
534	Authentication mechanism is too weak.
538	Encryption required for requested authentication mechanism.
550	Requested action not taken; mailbox unavailable.
551	User not local; please try <forward path>
552	Requested mail action aborted; exceeded storage allocation
553	Requested action not taken; mailbox name not allowed.
554	Transaction failed

ECEMAIL Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module.V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

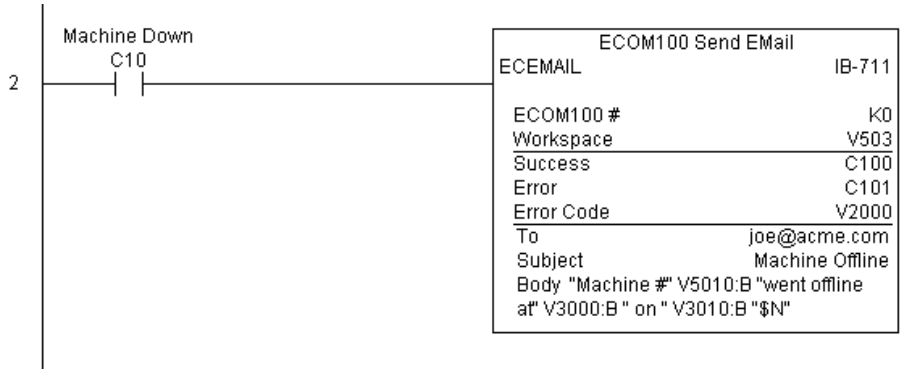
***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*



Rung 2: When a machine goes down, send an email to Joe in maintenance and to the VP over production showing what machine is down along with the date/time stamp of when it went down.

The ECEMAIL is leading edge triggered, not power-flow driven (similar to a counter input leg). An email will be sent whenever the power flow into the IBox goes from OFF to ON. This helps prevent self-inflicted spamming.

If the EMail is sent, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the SMTP error code or other possible error codes.

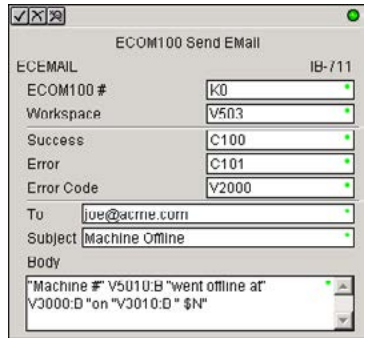


ECOM100 Restore Default E-mail Setup (ECEMRDS) (IB-713)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 Restore Default EMail Setup, on a leading edge transition, will restore the original EMail Setup data stored in the ECOM100 back to the working copy based on the specified ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) at the top of your program.



When the ECOM100 is first powered up, it copies the EMail setup data stored in ROM to the working copy in RAM. You can then modify this working copy from your program using the ECOM100 EMail Setup (ECEMSUP) IBox. After modifying the working copy, you can later restore the original setup data via your program by using this IBox.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

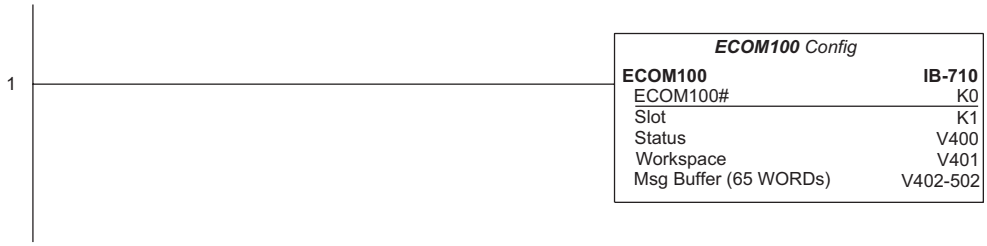
ECEMRDS Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Error Code: specifies the location where the Error Code will be written

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map - Data Words

ECEMRDS Example

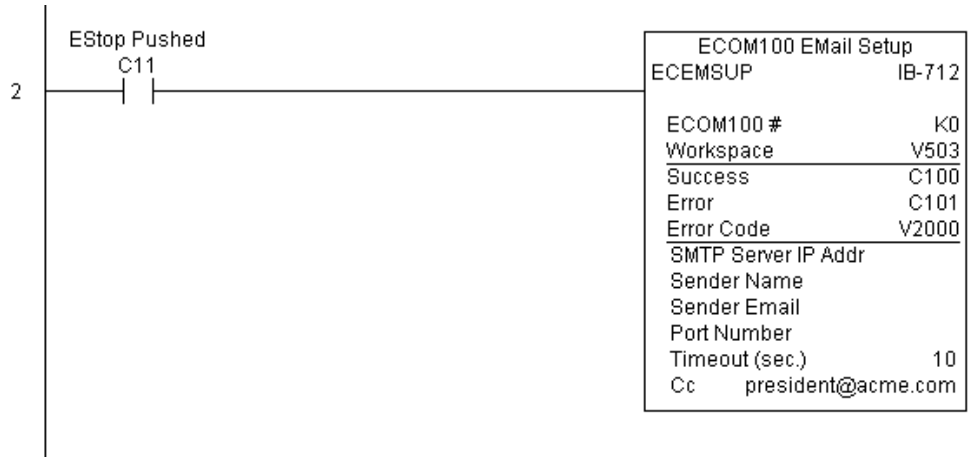
Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module.V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: Whenever an EStop is pushed, ensure that the president of the company gets copies of all Emails being sent.

The ECOM100 EMail Setup IBox allows you to set/change the SMTP Email settings stored in the ECOM100.



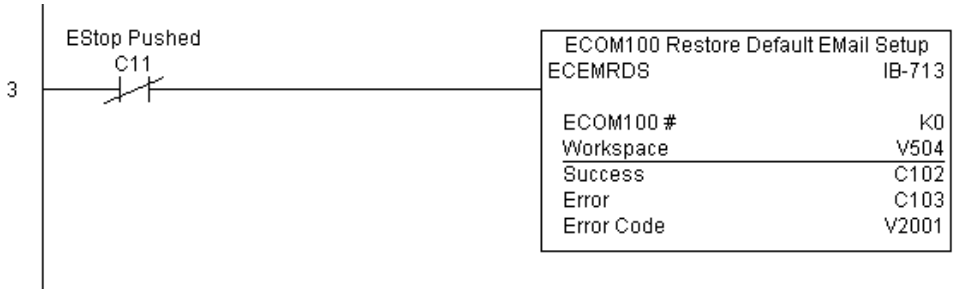
(Example continued on next page)

ECEMRDS Example (cont'd)

Rung 3: Once the EStop is pulled out, take the president off the cc: list by restoring the default EMail setup in the ECOM100.

The ECEMRDS is leading edge triggered, not power-flow driven (similar to a counter input leg). The ROM-based EMail configuration stored in the ECOM100 will be copied over the "working copy" whenever the power flow into the IBox goes from OFF to ON (the working copy can be changed by using the ECEMSUP IBox).

If successful, turn on C102. If there is a failure, turn on C103. If it fails, you can look at V2001 for the specific error code.

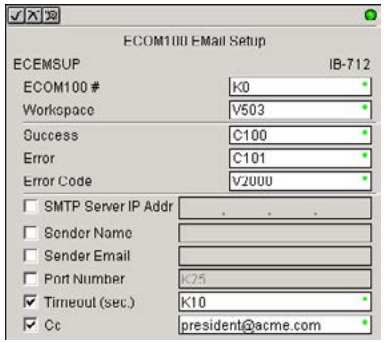


ECOM100 E-mail Setup (ECEMSUP) (IB-712)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 EMail Setup, on a leading edge transition, will modify the working copy of the EMail setup currently in the ECOM100 based on the specified ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) at the top of your program.



You may pick and choose any or all fields to be modified using this instruction. Note that these changes are cumulative: if you execute multiple ECOM100 EMail Setup IBoxes, then all of the changes are made in the order they are executed. Also note that you can restore the original ECOM100 EMail Setup that is stored in the ECOM100 to the working copy by using the ECOM100 Restore Default EMail Setup (ECEMRDS) IBox.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

You are limited to approximately 100 characters/bytes of setup data for the entire instruction. So if needed, you could divide the entire setup across multiple ECEMSUP IBoxes on a field-by-field basis, for example do the Carbon Copy (cc:) field in one ECEMSUP IBox and the remaining setup parameters in another.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECEMSUP Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Error Code: specifies the location where the Error Code will be written.
- SMTP Server IP Addr: optional parameter that specifies the IP Address of the SMTP Server on the ECOM100's network.
- Sender Name: optional parameter that specifies the sender name that will appear in the "From:" field to those who receive the e-mail.
- Sender EMail: optional parameter that specifies the sender EMail address that will appear in the "From:" field to those who receive the e-mail.

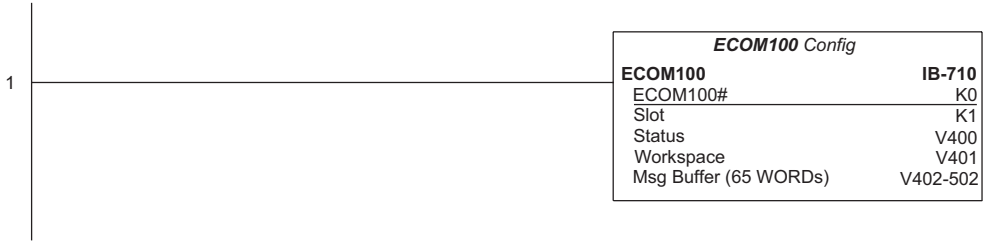
ECEMSUP Parameters (cont'd)

- Port Number: optional parameter that specifies the TCP/IP Port Number to send SMTP requests; usually this does not need to be configured (see your network administrator for information on this setting).
- Timeout (sec): optional parameter that specifies the number of seconds to wait for the SMTP Server to send the EMail to all the recipients.
- Cc: optional parameter that specifies a list of "carbon copy" Email addresses to send all EMails to.

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map - Data Words

ECEMSUP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module.V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



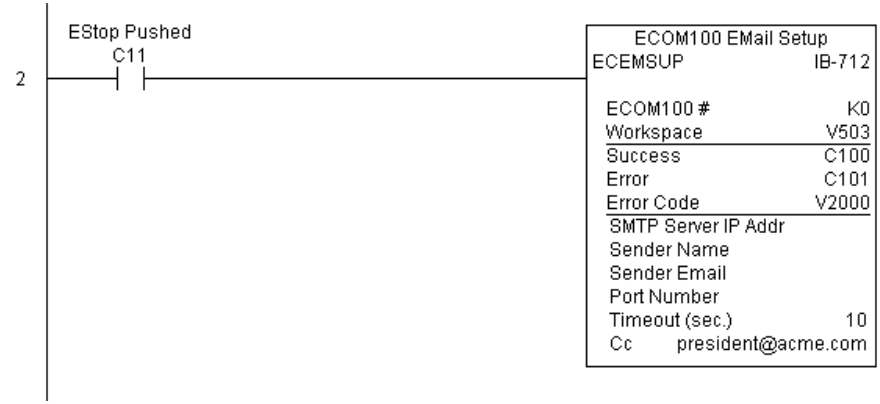
(Example continued on next page)

***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

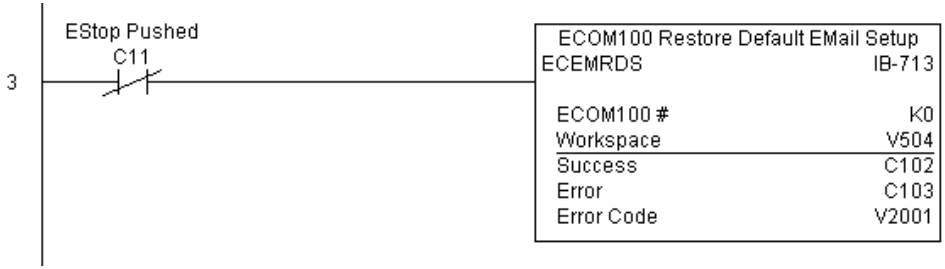
ECEMSUP Example (cont'd)

Rung 2: Whenever an EStop is pushed, ensure that president of the company gets copies of all EMails being sent.The ECOM100 EMail Setup IBox allows you to set/change the SMTP EMail settings stored in the ECOM100. The ECEMSUP is leading edge triggered, not power-flow driven (similar to a counter input leg). At power-up, the ROM-based EMail configuration stored in the ECOM100 is copied to a RAM-based "working copy". You can change this working copy by using the ECEMSUP IBox. To restore the original ROM-based configuration, use the Restore Default EMail Setup ECEMRDS IBox.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



Rung 3: Once the EStop is pulled out, take the president off the cc: list by restoring the default EMail setup in the ECOM100.

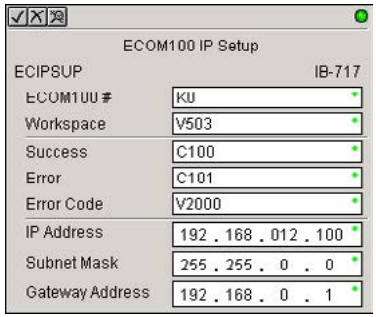


ECOM100 IP Setup (ECIPSUP) (IB-717)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 IP Setup will configure the three TCP/IP parameters in the ECOM100: IP Address, Subnet Mask, and Gateway Address, on a leading edge transition to the IBox. The ECOM100 is specified by the ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) IBox at the top of your program.



The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

This setup data is stored in Flash-ROM in the ECOM100 and will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is HIGHLY RECOMMENDED that you only execute this IBox ONCE on the second scan. Since it requires a LEADING edge to execute, use a NORMALLY CLOSED SP0 (NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECIPSUP Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Error Code: specifies the location where the Error Code will be written.
- IP Address: specifies the module's IP Address.
- Subnet Mask: specifies the Subnet Mask for the module to use.
- Gateway Address: specifies the Gateway Address for the module to use.

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map - Data Words
IP Address	IP Address	0.0.0.1. to 255.255.255.254
Subnet Mask Address	IP Address Mask	0.0.0.1. to 255.255.255.254
Gateway Address	IP Address	0.0.0.1. to 255.255.255.254

ECIPSUP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: On the second scan, configure all of the TCP/IP parameters in the ECOM100:

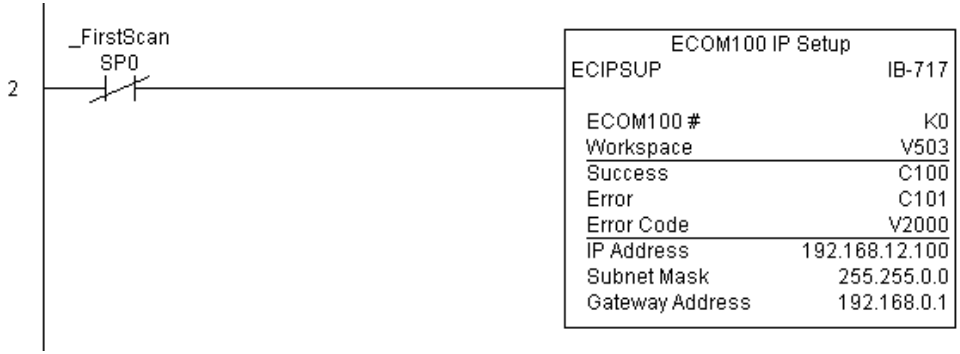
IP Address: 192.168.12.100

Subnet Mask: 255.255.0.0

Gateway Address: 192.168.0.1

The ECIPSUP is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the TCP/IP configuration parameters will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



ECOM100 Read Description (ECRDDES) (IB-726)

ECOM100 Read Description will read the ECOM100's Description field up to the number of specified characters on a leading edge transition to the IBox.

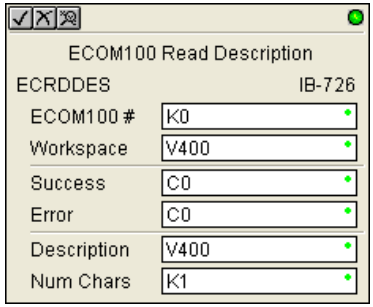
☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.



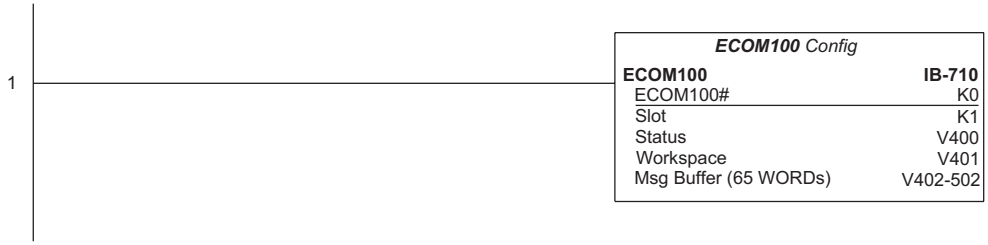
ECRDDES Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Description: specifies the starting buffer location where the ECOM100's Description will be placed
- Num Chars: specifies the number of characters (bytes) to read from the ECOM100's Description field

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Description	V	See DL205 V-memory map - Data Words
Num Chars	K	K1-128

ECRDDES Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: On the 2nd scan, read the Module Description of the ECOM100 and store it in V3000 thru V3007 (16 characters). This text can be displayed by an HMI.

The ECRDDES is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the module description will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.

ECOM100 Read Gateway Address (ECRDGWA) (IB-730)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

ECOM100 Read Gateway Address will read the four parts of the Gateway IP address and store them in four consecutive V-memory locations in decimal format, on a leading edge transition to the IBox.

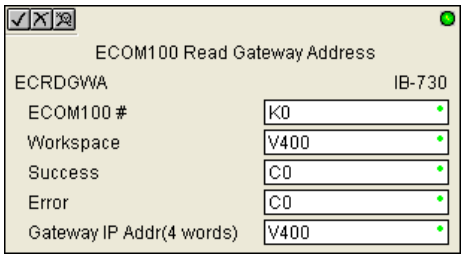
The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

DS5	Used
HPP	N/A

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECRDGWA Parameters

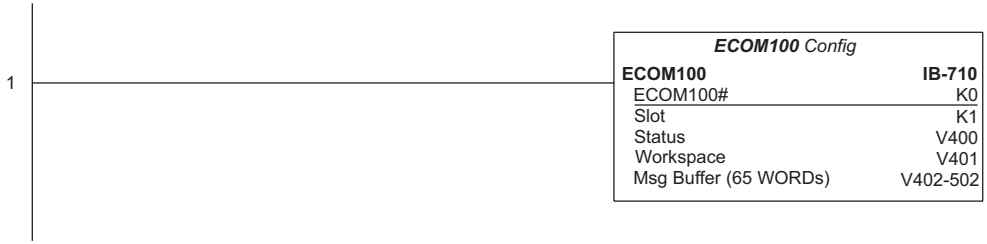


- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Gateway IP Addr: specifies the starting address where the ECOM100's Gateway Address will be placed in 4 consecutive V-memory locations.

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Gateway IP Address (4 Words)	V	See DL205 V-memory map - Data Words

ECRDGWA Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

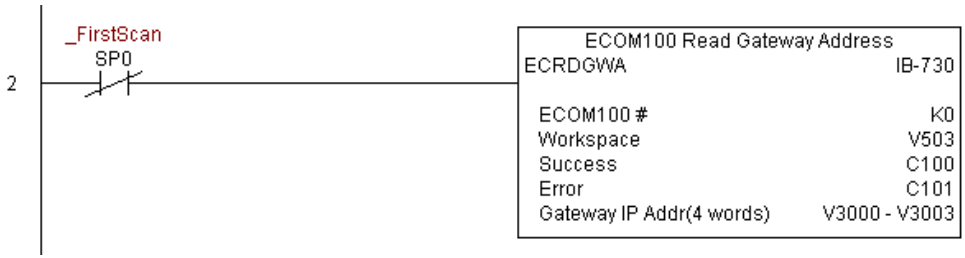


NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

Rung 2: On the second scan, read the Gateway Address of the ECOM100 and store it in V3000 thru V3003 (4 decimal numbers). The ECOM100's Gateway Address could be displayed by an HMI.

The ECRDGWA is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the Gateway Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



ECOM100 Read IP Address (ECRDIP) (IB-722)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

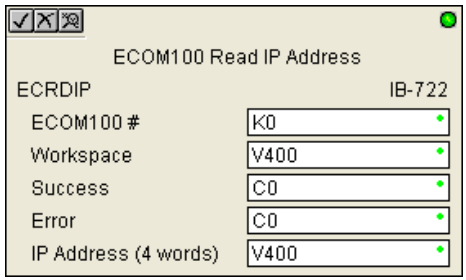
ECOM100 Read IP Address will read the four parts of the IP address and store them in four consecutive V-memory locations in decimal format, on a leading edge transition to the IBox.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECRDIP Parameters

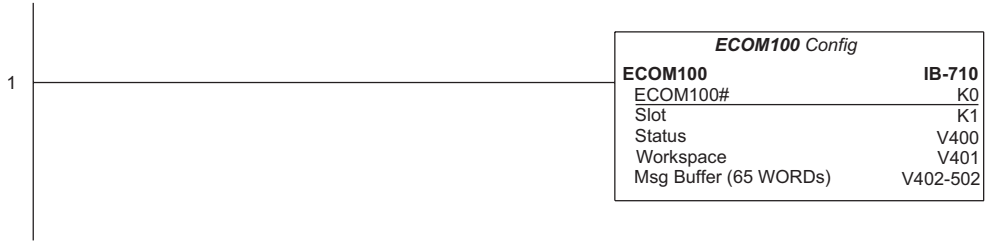


- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- IP Address: specifies the starting address where the ECOM100's IP Address will be placed in four consecutive V-memory locations

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
IP Address (4 Words)	V	See DL205 V-memory map - Data Words

ECRDIP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

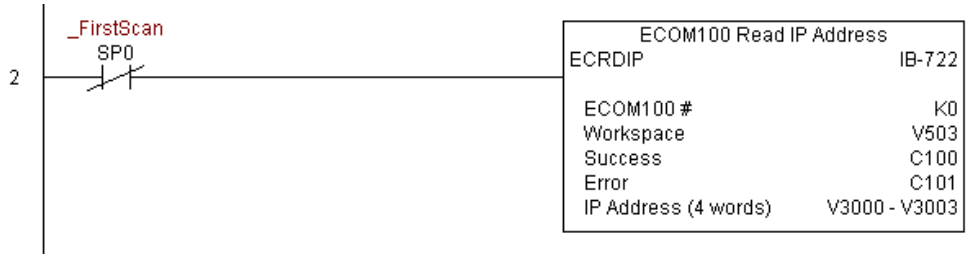


***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: On the second scan, read the IP Address of the ECOM100 and store it in V3000 thru V3003 (four decimal numbers). The ECOM100's IP Address could be displayed by an HMI.

The ECRDIP is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the IP Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



ECOM100 Read Module ID (ECRDMID) (IB-720)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

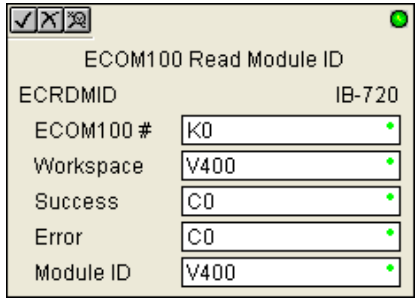
ECOM100 Read Module ID will read the binary (decimal) WORD sized Module ID on a leading edge transition to the IBox.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECRDMID Parameters



- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Module ID: specifies the location where the ECOM100's Module ID (decimal) will be placed

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Module ID	V	See DL205 V-memory map - Data Words

ECRDMID Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

Rung 2: On the second scan, read the Module ID of the ECOM100 and store it in V2000.

The ECRDMID is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the module ID will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.

ECOM100 Read Module Name (ECRDNAM) (IB-724)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

ECOM100 Read Name will read the Module Name up to the number of specified characters on a leading edge transition to the IBox.

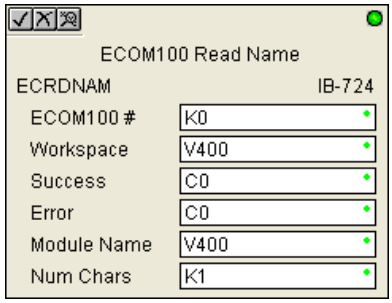
The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

DS5	Used
HPP	N/A

ECRDNAM Parameters



- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Module Name: specifies the starting buffer location where the ECOM100's Module Name will be placed.
- Num Chars: specifies the number of characters (bytes) to read from the ECOM100's Name field.

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Module Name	V	See DL205 V-memory map - Data Words
Num Chars	K	K1-128

ECRDNAM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

Rung 2: On the second scan, read the Module Name of the ECOM100 and store it in V3000 thru V3003 (8 characters). This text can be displayed by an HMI.

The ECRDNAM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the module name will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.

ECOM100 Read Subnet Mask (ECRDSNM) (IB-732)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

ECOM100 Read Subnet Mask will read the four parts of the Subnet Mask and store them in 4 consecutive V-memory locations in decimal format, on a leading edge transition to the IBox.

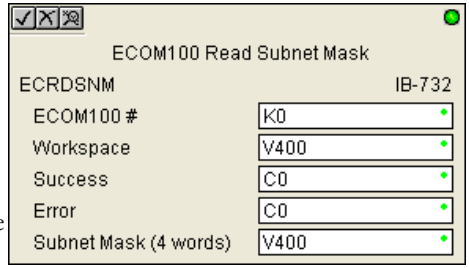
The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

DS5	Used
HPP	N/A

ECRDSNM Parameters

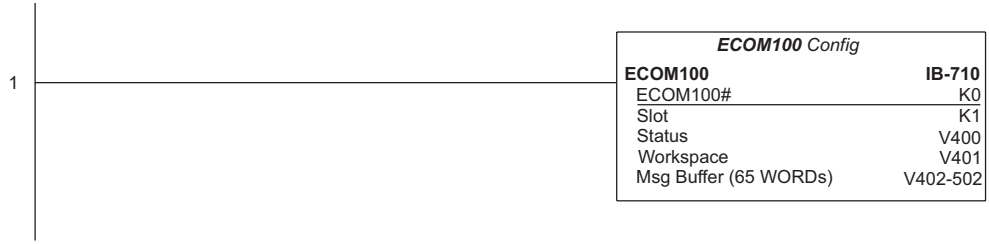


- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Subnet Mask: specifies the starting address where the ECOM100's Subnet Mask will be placed in four consecutive V-memory locations.

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Subnet Mask (4 Words)	V	See DL205 V-memory map - Data Words

ECRDSNM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

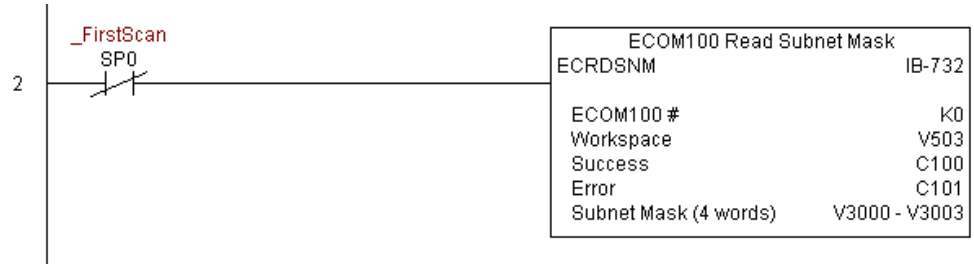


***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: On the second scan, read the Subnet Mask of the ECOM100 and store it in V3000 thru V3003 (4 decimal numbers). The ECOM100's Subnet Mask could be displayed by an HMI.

The ECRDSNM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the Subnet Mask will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



ECOM100 Write Description (ECWRDES) (IB-727)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 Write Description will write the given Description to the ECOM100 module on a leading edge transition to the IBox. If you use a dollar sign ($) or double quote (“), use the PRINT/VPRINT escape sequence of TWO dollar signs ($$) for a single dollar sign or dollar sign-double quote ($”) for a double quote character.

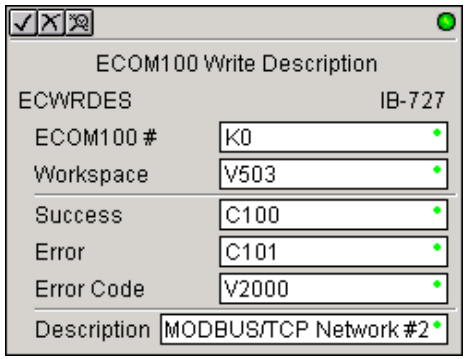
The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Description is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is HIGHLY RECOMMENDED that you only execute this IBox ONCE on the second scan. Since it requires a LEADING edge to execute, use a NORMALLY CLOSED SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECWRDES Parameters

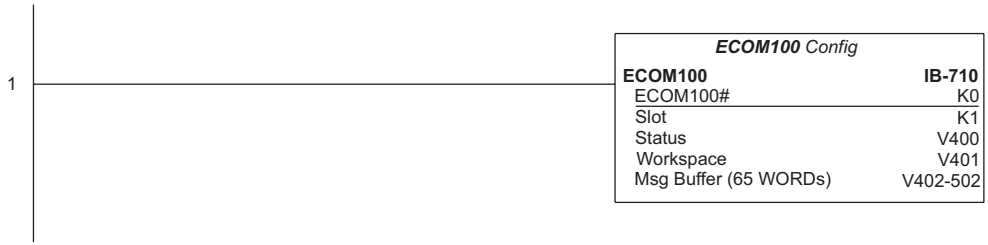


- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Error Code: specifies the location where the Error Code will be written
- Description: specifies the Description that will be written to the module

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map - Data Words
Description		Text

ECWRDES Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

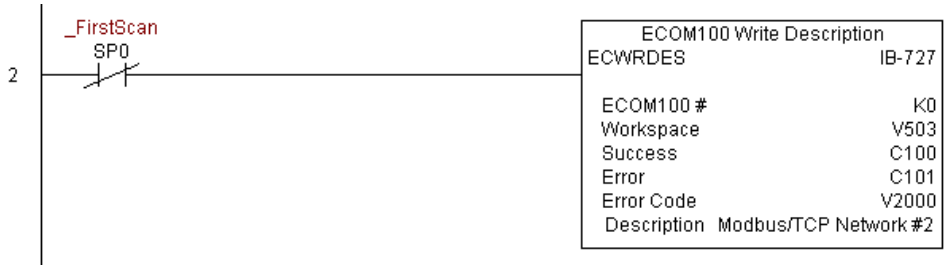


NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

Rung 2: On the second scan, set the Module Description of the ECOM100. Typically this is done using NetEdit, but this IBox allows you to configure the module description in the ECOM100 using your ladder program.

The ECWRDES is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the module description will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



ECOM100 Write Gateway Address (ECWRGWA) (IB-731)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 Write Gateway Address will write the given Gateway IP Address to the ECOM100 module on a leading edge transition to the IBox. See also ECOM100 IP Setup (ECIPSUP) IBox 717 to set up ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.

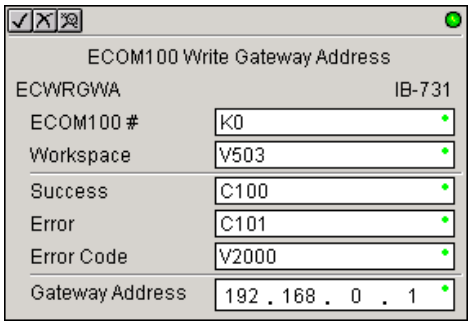
The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Gateway Address is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is HIGHLY RECOMMENDED that you only execute this IBox ONCE, on the second scan. Since it requires a LEADING edge to execute, use a NORMALLY CLOSED SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECWRGWA Parameters



- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully
- Error Code: specifies the location where the Error Code will be written
- Gateway Address: specifies the Gateway IP Address that will be written to the module

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map - Data Words
Gateway Address		0.0.0.1. to 255.255.255.254

ECWRGWA Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

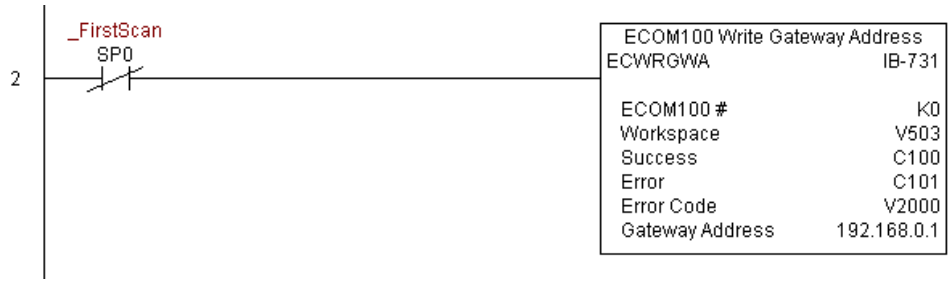
NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

Rung 2: On the second scan, assign the Gateway Address of the ECOM100 to 192.168.0.1

The ECWRGWA is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the Gateway Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

To configure all of the ECOM100 TCP/IP parameters in one IBox, see the ECOM100 IP Setup (ECIPSUP) IBox.



ECOM100 Write IP Address (ECWRIP) (IB-723)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 Write IP Address will write the given IP Address to the ECOM100 module on a leading edge transition to the IBox. See also ECOM100 IP Setup (ECIPSUP) IBox 717 to setup ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.

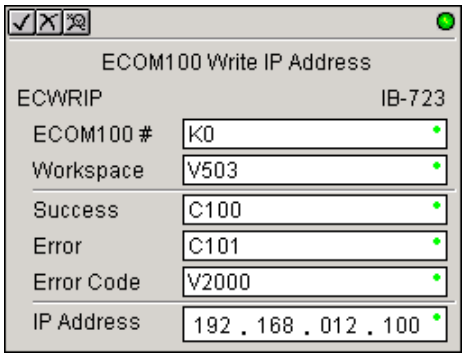
The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The IP Address is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is HIGHLY RECOMMENDED that you only execute this IBox ONCE on the second scan. Since it requires a LEADING edge to execute, use a NORMALLY CLOSED SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECWRIP Parameters

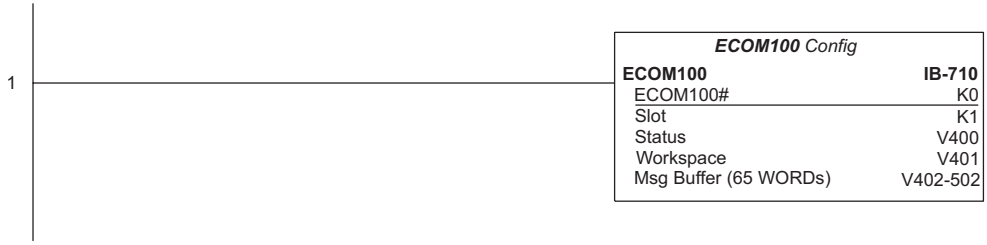


- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written
- IP Address: specifies the IP Address that will be written to the module

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map - Data Words
IP Address		0.0.0.1. to 255.255.255.254

ECWRIP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

Rung 2: On the second scan, assign the IP Address of the ECOM100 to 192.168.12.100

The ECWRIP is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the IP Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

To configure all of the ECOM100 TCP/IP parameters in one IBox, see the ECOM100 IP Setup (ECIPSUP) IBox.

ECOM100 Write Module ID (ECWRMID) (IB-721)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 Write Module ID will write the given Module ID on a leading edge transition to the IBox

If the Module ID is set in the hardware using the dipswitches, this IBox will fail and return error code 1005 (decimal).

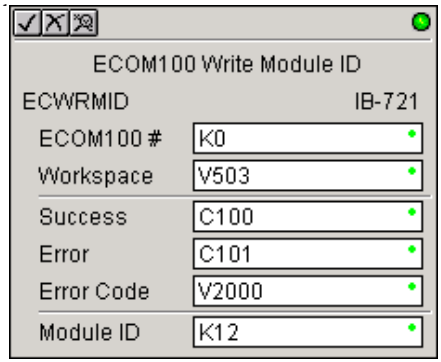
The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Module ID is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is HIGHLY RECOMMENDED that you only execute this IBox ONCE on the second scan. Since it requires a LEADING edge to execute, use a NORMALLY CLOSED SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECWRMID Parameters

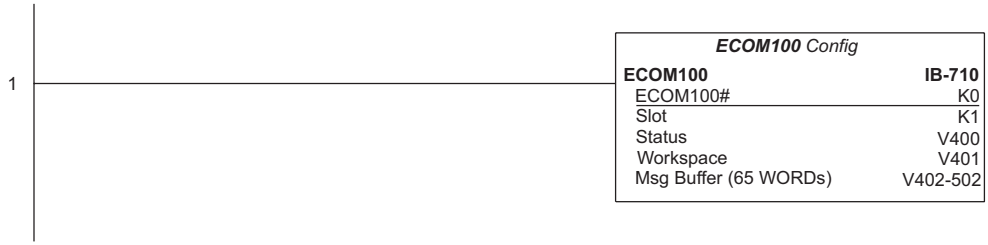


- ECOM100#: this is a logical number associate with this specific ECOM100 module in th specified slot. All other ECxxxx IBoxes tha need to reference this ECOM100 module mu reference this logical number
- Workspace: specifies a V-memory location tha will be used by the instruction
- Success: specifies a bit that will turn on once th request is completed successfully
- Error: specifies a bit that will turn on if th instruction is not completed successfully
- Error Code: specifies the location where th Error Code will be written
- Module ID: specifies the Module ID that will be written to the module

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map - Data Words
Module ID		K0-65535

ECWRMID Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: On the second scan, set the Module ID of the ECOM100. Typically this is done using NetEdit, but this IBox allows you to configure the module ID of the ECOM100 using your ladder program.

The ECWRMID is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the module ID will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

ECOM100 Write Name (ECWRNAM) (IB-725)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 Write Name will write the given Name to the ECOM100 module on a leading edge transition to the IBox. If you use a dollar sign ($) or double quote (“), use the PRINT/VPRINT escape sequence of TWO dollar signs ($$) for a single dollar sign or dollar sign-double quote ($”) for a double quote character.

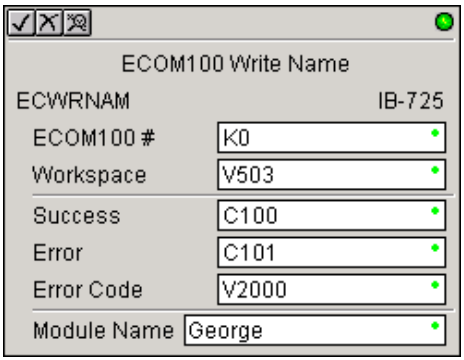
The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Name is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is HIGHLY RECOMMENDED that you only execute this IBox ONCE on the second scan. Since it requires a LEADING edge to execute, use a NORMALLY CLOSED SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECWRNAM Parameters

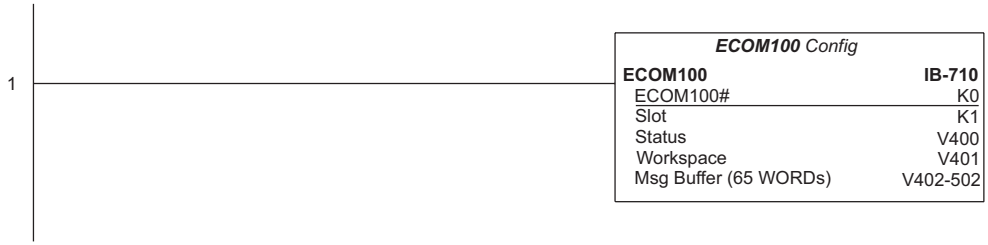


- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Error Code: specifies the location where the Error Code will be written.
- Module Name: specifies the Name that will be written to the module.

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map - Data Words
Module Name		Text

ECWRNAM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: On the second scan, set the Module Name of the ECOM100. Typically this is done using NetEdit, but this IBox allows you to configure the module name of the ECOM100 using your ladder program.

The ECWRNAM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the module name will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

ECOM100 Write Subnet Mask (ECWRSNM) (IB-733)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

ECOM100 Write Subnet Mask will write the given Subnet Mask to the ECOM100 module on a leading edge transition to the IBox. See also ECOM100 IP Setup (ECIPSUP) IBox 717 to set up ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.

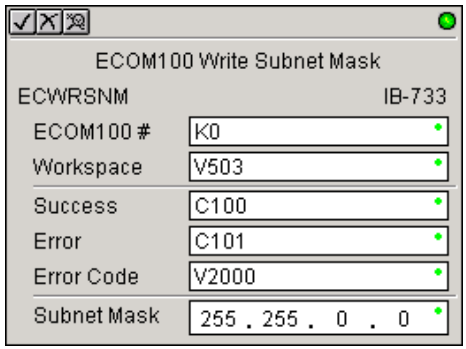
The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Subnet Mask is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is HIGHLY RECOMMENDED that you only execute this IBox ONCE on the second scan. Since it requires a LEADING edge to execute, use a NORMALLY CLOSED SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

ECWRSNM Parameters



- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.
- Error Code: specifies the location where the Error Code will be written.
- Subnet Mask: specifies the Subnet Mask that will be written to the module.

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map
Error Code	V	See DL205 V-memory map - Data Words
Subnet Mask		Masked IP Address

ECWRSNM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

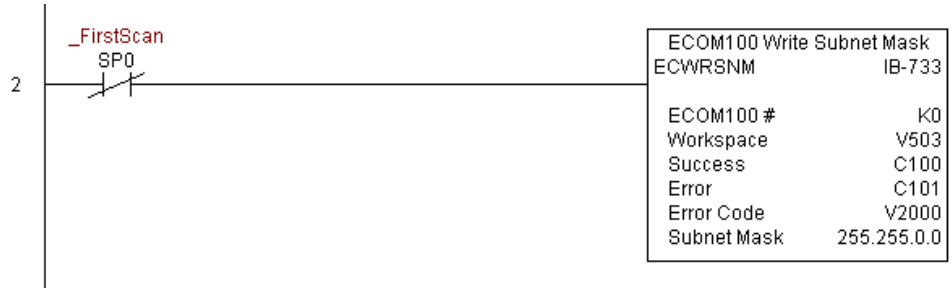
NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

Rung 2: On the second scan, assign the Subnet Mask of the ECOM100 to 255.255.0.0

The ECWRSNM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the Subnet Mask will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

To configure all of the ECOM100 TCP/IP parameters in one IBox, see the ECOM100 IP Setup (ECIPSUP) IBox.



ECOM100 RX Network Read (ECRX) (IB-740)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

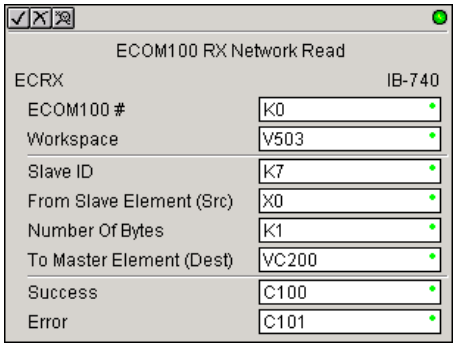
ECOM100 RX Network Read performs the RX instruction with built-in interlocking with all other ECOM100 RX (ECRX) and ECOM100 WX (ECWX) IBoxes in your program to simplify communications networking. It will perform the RX on the specified ECOM100#'s network, which corresponds to a specific unique ECOM100 Configuration (ECOM100) IBox at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Whenever this IBox has power, it will read element data from the specified slave into the given destination V-memory buffer, giving other ECOM100 RX and ECOM100 WX IBoxes on that ECOM100# network a chance to execute.

For example, if you wish to read and write data continuously from five different slaves, you can have all of these ECRX and ECWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically!

ECRX Parameters

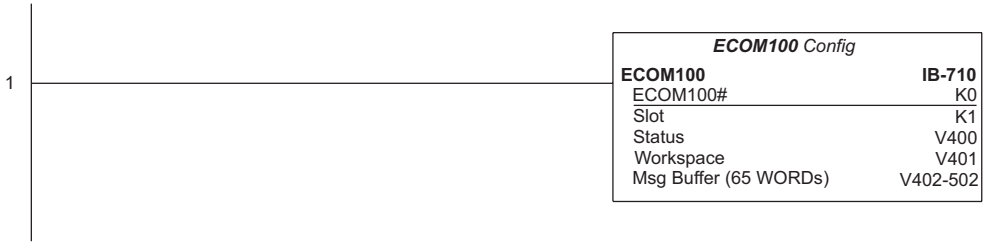


- ECOM100#: this is a logical numb associated with this specific ECOM1(module in the specified slot. All other ECxx IBoxes that need to reference this ECOM1(module must reference this logical number
- Workspace: specifies a V-memory locatic that will be used by the instruction
- Slave ID: specifies the slave ECOM(100) PL that will be targeted by the ECRX instructic
- From Slave Element (Src): specifies the sla address of the data to be read
- Number of Bytes: specifies the number bytes to read from the slave ECOM(100) PLC
- To Master Element (Dest): specifies the location where the slave data will be placed in the master ECOM100 PLC
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Slave ID	K	K0-90
From Slave Element (Src)	X,Y,C,S,T,CT,GX,GY,V	See DL205 V-memory map
Number of Bytes	K	K1-128
To Master Element (Dest)	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

ECRX Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



(Example continued on next page)

NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

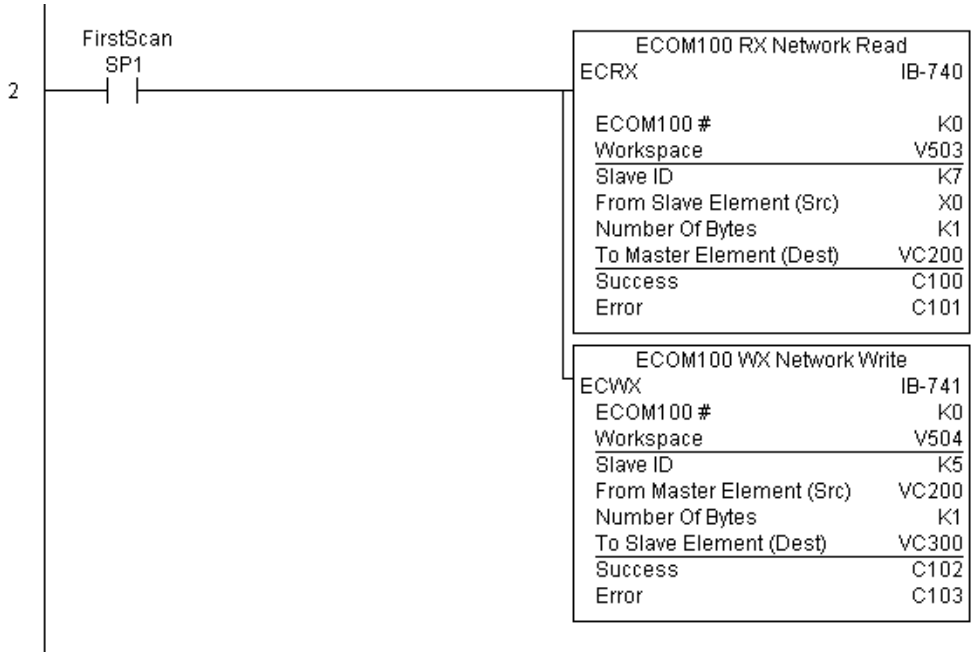
ECRX Example (cont'd)

Rung 2: Using ECOM100# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the ECRX and ECWX work with the ECOM100 Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP "busy bits" or "error bits," or what slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the ECRX and ECWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending ECOM100 IBoxes below the ECWX, then the very next scan the ECRX would start its request again.

Using the ECRX and ECWX for all of your ECOM100 network reads and writes is the fastest the PLC can do networking. For local Serial Ports, DCM modules, or the original ECOM modules, use the NETCFG and NETRX/NETWX IBoxes.



ECOM100 WX Network Write(ECWX) (IB-741)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

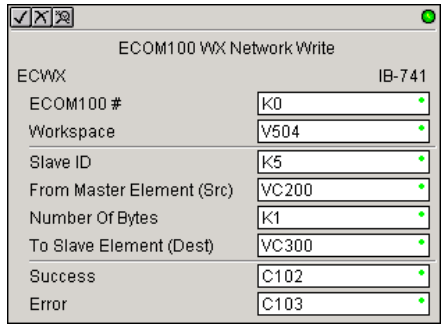
ECOM100 WX Network Write performs the WX instruction with built-in interlocking with all other ECOM100 RX (ECRX) and ECOM100 WX (ECWX) IBoxes in your program to simplify communications networking. It will perform the WX on the specified ECOM100#'s network, which corresponds to a specific unique ECOM100 Configuration (ECOM100) IBox at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Whenever this IBox has power, it will write data from the master's V-memory buffer to the specified slave starting with the given slave element, giving other ECOM100 RX and ECOM100 WX IBoxes on that ECOM100# network a chance to execute.

For example, if you wish to read and write data continuously from five different slaves, you can have all of these ECRX and ECWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically!

ECWX Parameters

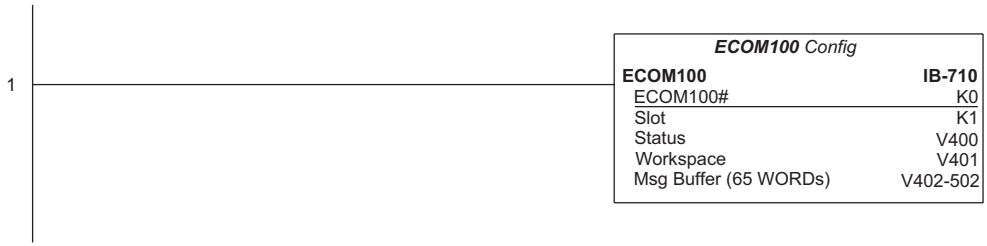


- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Slave ID: specifies the slave ECOM(100) PLC that will be targeted by the ECWX instruction.
- From Master Element (Src): specifies the location in the master ECOM100 PLC where the data will be sourced from.
- Number of Bytes: specifies the number of bytes to write to the slave ECOM(100) PLC.
- To Slave Element (Dest): specifies the slave address the data will be written to.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.

Parameter		DL205 Range
ECOM100#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Slave ID	K	K0-90
From Master Element (Src)	V	See DL205 V-memory map - Data Words
Number of Bytes	K	K1-128
To Slave Element (Dest)	X,Y,C,S,T,CT,GX,GY,V	See DL205 V-memory map
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

ECWX Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module number as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130-byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



***NOTE**: An ECOM100 IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

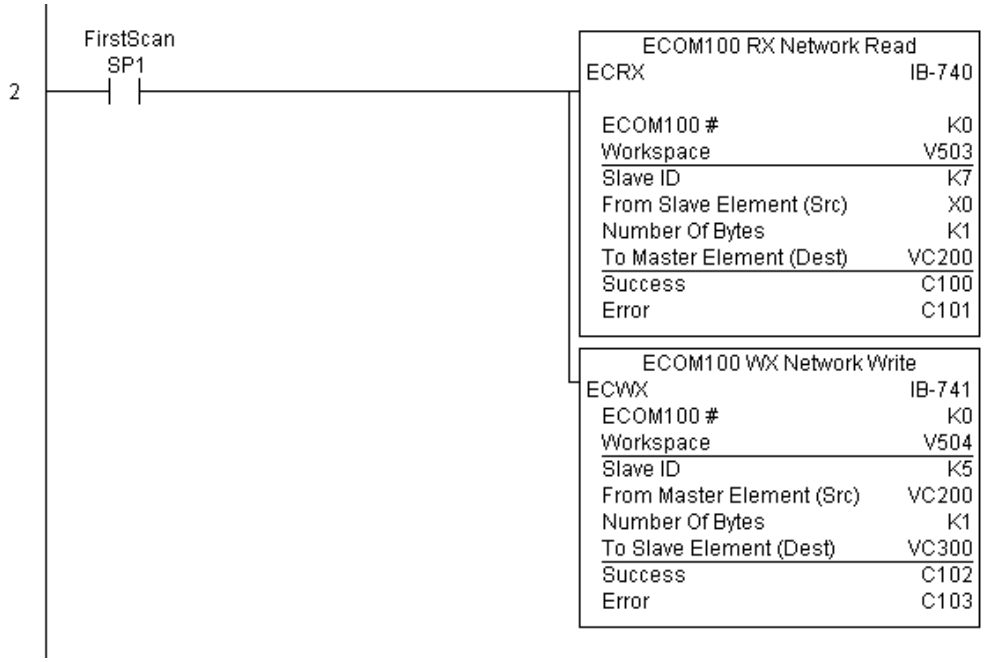
ECWX Example (cont'd)

Rung 2: Using ECOM100# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the ECRX and ECWX work with the ECOM100 Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP "busy bits" or "error bits," or what slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the ECRX and ECWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending ECOM100 IBoxes below the ECWX, then the very next scan the ECRX would start its request again.

Using the ECRX and ECWX for all of your ECOM100 network reads and writes is the fastest the PLC can do networking. For local Serial Ports, DCM modules, or the original ECOM modules, use the NETCFG and NETRX/NETWX IBoxes.



NETCFG Network Configuration (NETCFG) (IB-700)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

Network Config defines all the common information necessary for performing RX/WX Networking using the NETRX and NETWX IBox instructions via a local CPU serial port, DCM or ECOM module.

You must have the Network Config instruction at the top of your ladder/stage program with any other configuration IBoxes.

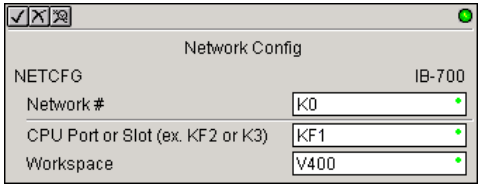
If you use more than one local serial port, DCM or ECOM in your PLC for RX/WX Networking, you must have a different Network Config instruction and Network number for EACH RX/WX network in your system that utilizes any NETRX/NETWX IBox instructions.

The second parameter "CPU Port or Slot" is the same value as in the high byte of the first LD instruction if you were coding the RX or WX rung yourself. This value is CPU and port specific. Use KF1 for local CPU serial port 2. Use K3 if a DCM or ECOM is located in slot 3 of a local 205 base.

The Workspace parameter is an internal, private register used by the Network Config IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

NETCFG Parameters

- Network#: specifies a unique number for each ECOM(100) or DCM network to use
- CPU Port or Slot: specifies the CPU port number or slot number of DCM/ECOM(100) used
- Workspace: specifies a V-memory location that will be used by the instruction



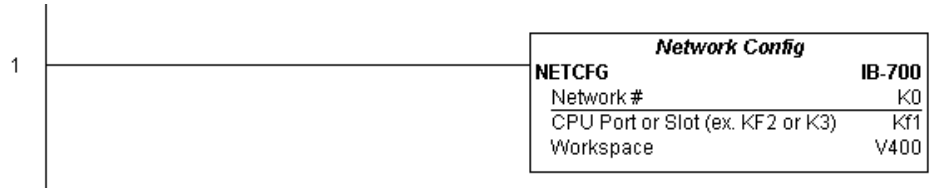
Parameter		DL205 Range
Network#	K	K0-255
CPU Port or Slot	K	K0-FF
Workspace	V	See DL205 V-memory map - Data Words

NETCFG Example

The Network Configuration IBox coordinates all of the interaction with other Network IBoxes (NETRX/NETWX). You must have a Network Configuration IBox for each serial port network, DCM module network, or original ECOM module network in your system. Configuration IBoxes must be at the top of your program and must execute every scan.

This IBox defines Network# K0 to be for the local CPU serial port #2 (KF1). For local CPU serial ports or DCM/ECOM modules, use the same value you would use in the most significant byte of the first LD instruction in a normal RX/WX rung to reference the port or module. Any NETRX or NETWX IBoxes that need to reference this specific network would enter K0 for their Network# parameter.

The Workspace register is used to maintain state information about the port or module, along with proper sharing and interlocking with the other NETRX and NETWX IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.



***NOTE**: The Network Configuration IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Network RX Read (NETRX) (IB-701)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

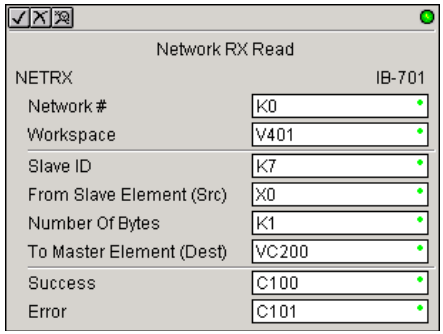
Network RX Read performs the RX instruction with built-in interlocking with all other Network RX (NETRX) and Network WX (NETWX) IBoxes in your program to simplify communications networking. It will perform the RX on the specified Network number, which corresponds to a specific unique Network Configuration (NETCFG) at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Whenever this IBox has power, it will read element data from the specified slave into the given destination V-memory buffer, giving other Network RX and Network WX IBoxes on that Network number a chance to execute.

For example, if you wish to read and write data continuously from five different slaves, you can have all of these NETRX and NETWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically!

NETRX Parameters



- Network#: specifies the (CPU ports, DCMs, ECOMs) Network # defined by the NETCFG instruction.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Slave ID: specifies the slave PLC that will be targeted by the NETRX instruction.
- From Slave Element (Src): specifies the slave address of the data to be read.
- Number of Bytes: specifies the number of bytes to read from the slave device.
- To Master Element (Dest): specifies the location where the slave data will be placed in the master PLC.
- Success: specifies a bit that will turn on once the request is completed successfully.
- Error: specifies a bit that will turn on if the instruction is not completed successfully.

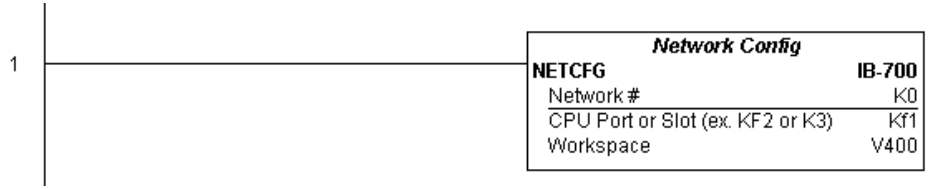
Parameter		DL205 Range
Network#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Slave ID	K,V	K0-90
From Slave Element (Src)	X,Y,C,S,T,CT,GX,GY,V	See DL205 V-memory map
Number of Bytes	K	K1-128
To Master Element (Dest)	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

NETRX Example

Rung 1: The Network Configuration IBox coordinates all of the interaction with other Network IBoxes (NETRX/NETWX). You must have a Network Configuration IBox for each serial port network, DCM module network, or original ECOM module network in your system. Configuration IBoxes must be at the top of your program and must execute every scan.

This IBox defines Network# K0 to be for the local CPU serial port #2 (KF1). For local CPU serial ports or DCM/ECOM modules, use the same value you would use in the most significant byte of the first LD instruction in a normal RX/WX rung to reference the port or module. Any NETRX or NETWX IBoxes that need to reference this specific network would enter K0 for their Network# parameter.

The Workspace register is used to maintain state information about the port or module, along with proper sharing and interlocking with the other NETRX and NETWX IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.



(Example continued on next page)

NOTE**: The Network Configuration IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

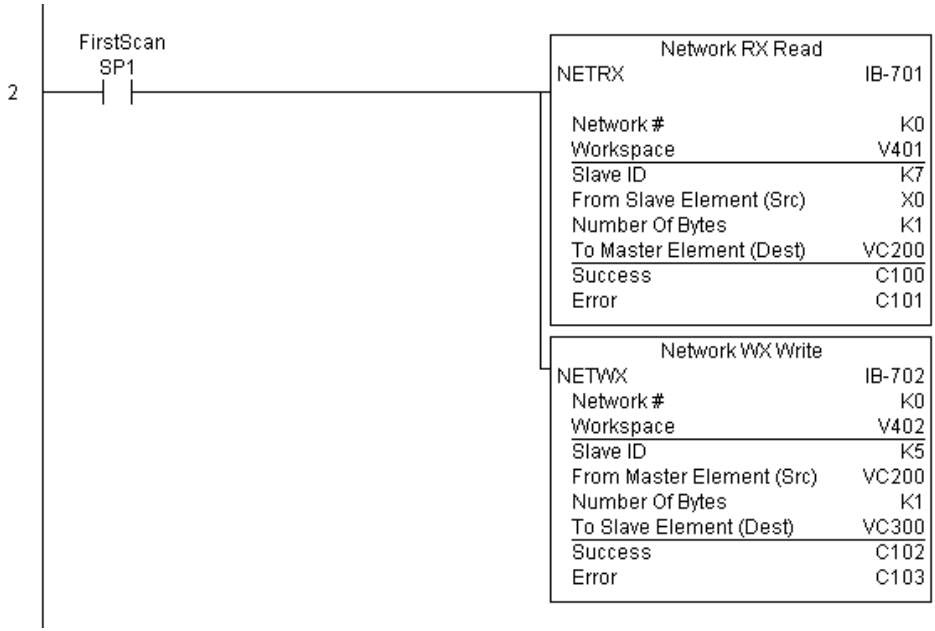
NETRX Example (cont'd)

Rung 2: Using Network# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the NETRX and NETWX work with the Network Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP "busy bits" or "error bits," or what port number or slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the NETRX and NETWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending NETRX or NETWX IBoxes below this IBox, then the very next scan the NETRX would start its request again.

Using the NETRX and NETWX for all of your serial port, DCM, or original ECOM network reads and writes is the fastest the PLC can do networking. For ECOM100 modules, use the ECOM100 and ECRX/ECWX IBoxes.



Network WX Write (NETWX) (IB-702)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

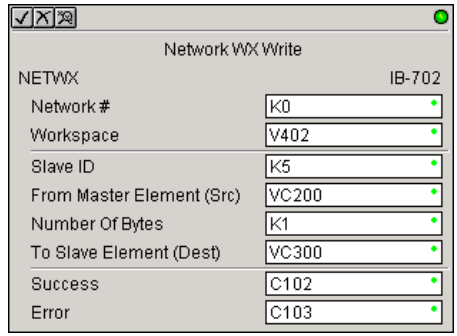
Network WX Write performs the WX instruction with built-in interlocking with all other Network RX (NETRX) and Network WX (NETWX) IBoxes in your program to simplify communications networking. It will perform the WX on the specified Network number, which corresponds to a specific unique Network Configuration (NETCFG) at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Whenever this IBox has power, it will write data from the master's V-memory buffer to the specified slave starting with the given slave element, giving other Network RX and Network WX IBoxes on that Network number a chance to execute.

For example, if you wish to read and write data continuously from five different slaves, you can have all of these NETRX and NETWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically!

NETWX Parameters



- Network#: specifies the (CPU ports, DCMs, ECOMs) Network # defined by the NETCFG instruction
- Workspace: specifies a V-memory location that will be used by the instruction
- Slave ID: specifies the slave PLC that will be targeted by the NETWX instruction
- From Master Element (Src): specifies the location in the master PLC where the data will be sourced
- Number of Bytes: specifies the number of bytes to write to the slave PLC
- To Slave Element (Dest): specifies the slave address the data will be written to
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not completed successfully

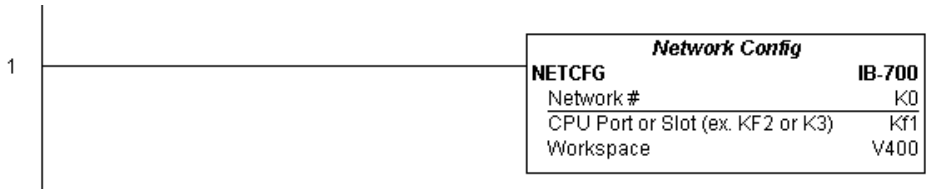
Parameter		DL205 Range
Network#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Slave ID	K,V	K0-90
From Master Element (Src)	V	See DL205 V-memory map - Data Words
Number of Bytes	K	K1-128
To Slave Element (Dest)	X,Y,C,S,T,CT,GX,GY,V	See DL205 V-memory map
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

NETWX Example

Rung 1: The Network Configuration IBox coordinates all of the interaction with other Network IBoxes (NETRX/NETWX). You must have a Network Configuration IBox for each serial port network, DCM module network, or original ECOM module network in your system. Configuration IBoxes must be at the top of your program and must execute every scan.

This IBox defines Network# K0 to be for the local CPU serial port #2 (KF1). For local CPU serial ports or DCM/ECOM modules, use the same value you would use in the most significant byte of the first LD instruction in a normal RX/WX rung to reference the port or module. Any NETRX or NETWX IBoxes that need to reference this specific network would enter K0 for their Network# parameter.

The Workspace register is used to maintain state information about the port or module, along with proper sharing and interlocking with the other NETRX and NETWX IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.



(Example continued on next page)

***NOTE**: The Network Configuration IBox instruction is used without a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

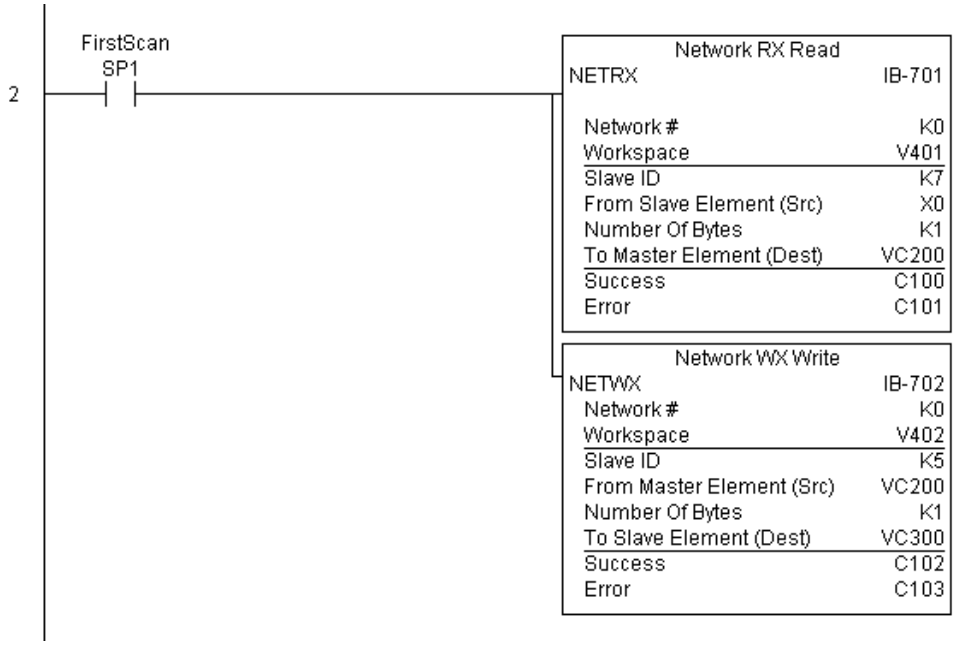
NETWX Example (cont'd)

Rung 2: Using Network# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the NETRX and NETWX work with the Network Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP "busy bits" or "error bits", or what port number or slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the NETRX and NETWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending NETRX or NETWX IBoxes below this IBox, then the very next scan the NETRX would start its request again.

Using the NETRX and NETWX for all of your serial port, DCM, or original ECOM network reads and writes is the fastest the PLC can do networking. For ECOM100 modules, use the ECOM100 and ECRX/ECWX IBoxes.



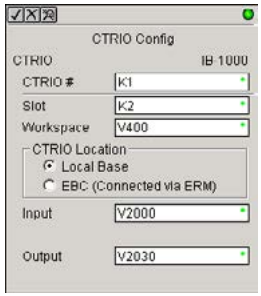
CTRIO Configuration (CTRIO) (IB-1000)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

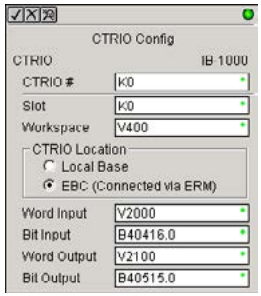
DS5	Used
HPP	N/A

CTRIO Config defines all the common information for one specific CTRIO module which is used by the other CTRIO IBox instructions (for example, CTRLDPR - CTRIO Load Profile, CTREDRL - CTRIO Edit and Reload Preset Table, CTRRTLM - CTRIO Run to Limit Mode, ...).

The Input/Output parameters for this instruction can be copied directly from the CTRIO Workbench configuration for this CTRIO module. Since the behavior is slightly different when the CTRIO module is in an EBC Base via an ERM, you must specify whether the CTRIO module is in a local base or in an EBC base.



CTRIO in Local Base



CTRIO in EBC Base

You must have the CTRIO Config IBox at the top of your ladder/stage program along with any other configuration IBoxes. If you have more than one CTRIO in your PLC, you must have a different CTRIO Config IBox for EACH CTRIO module in your system that utilizes any CTRIO IBox instructions. Each CTRIO Config IBox must have a UNIQUE CTRIO# value. This is how the CTRIO IBoxes differentiate between the different CTRIO modules in your system.

The Workspace parameter is an internal, private register used by the CTRIO Config IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

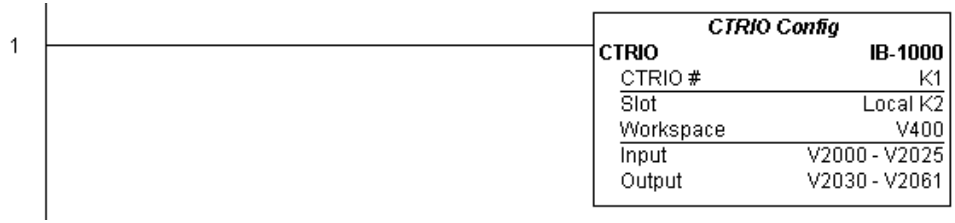
CTRIO Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number
- Slot: (local base): specifies which PLC slot is occupied by the module (always K0 for EBC base)
- Workspace: specifies a V-memory location that will be used by the instruction
- CTRIO Location: specifies where the module is located (PLC local base or ERM to EBC base)
- Input (local base): This needs to be set to the same V-memory register as is specified in CTRIO Workbench as 'Starting V address for inputs' for this unique CTRIO.
- Output (local base): This needs to be set to the same V-memory register as is specified in CTRIO Workbench as 'Starting V address for outputs' for this unique CTRIO.
- Word Input (EBC base): The starting input V-memory address as defined by the I/O configuration in the ERM Workbench
- Bit Input (EBC base): The starting input Bit address as defined by the I/O configuration in the ERM Workbench
- Word Output (EBC base): The starting output V-memory address as defined by the I/O configuration in the ERM Workbench
- Bit Output (EBC base): The starting output Bit address as defined by the I/O configuration in the ERM Workbench.

Parameter		DL205 Range
CTRIO#	K	K0-255
Slot	K	K0-7
Workspace	V	See DL205 V-memory map - Data Words
Input (Word, Bit)	V,B	See DL205 V-memory map - Data Words
Output (Word, Bit)	V,B	See DL205 V-memory map - Data Words

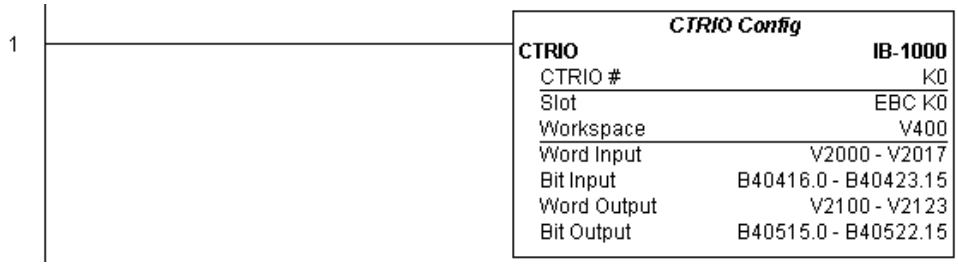
CTRIO Example (local base)

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



CTRIO Example (EBC base)

Overview: ERM Workbench must first be used to assign memory addresses to the I/O modules in the EBC base. Once the CTRIO module memory addresses are established using ERM Workbench, they are used in CTRIO Workbench and in a CTRIO IBox instruction to configure and define a specific CTRIO module. For this example, the CTRIO module uses V2000 - V2017 for its Word Input data and B40416.0 - B40423.15 for its Bit Input data. The module uses V2100 - V2123 for its Word Output data and B40515.0 - B40522.15 for its Bit Output data. The starting addresses, V2000 and V40416 (for inputs) and V2100 and V40515 (for outputs) are entered into CTRIO Workbench I/O Map to configure this specific CTRIO module. These starting addresses are the memory locations used in the CTRIO IBox instruction as the Word Input, Bit Input, Word Output and Bit Output addresses as shown below. For more information on this topic, refer to the CTRIO User Manual "Program Control" chapter.



***NOTE**: The CTRIO Configuration IBox instructions do not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD** characters.*

CTRIO Add Entry to End of Preset Table (CTRADPT) (IB-1005)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

CTRIO Add Entry to End of Preset Table, on a leading edge transition to this IBox, will append an entry to the end of a memory based Preset Table on a specific CTRIO Output resource. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

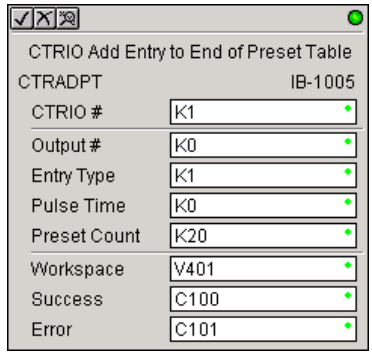
K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

CTRADPT Parameters



- CTRIO#: specifies a specific CTRIO module based on a user-defined number (see CTRIO Config)
- Output#: specifies a CTRIO output to be used by the instruction
- Entry Type: specifies the Entry Type to be added to the end of a Preset Table
- Pulse Time: specifies a pulse time in msecs for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter		DL205 Range
CTRIO#	K	K0-255
Output#	K	K0-3
Entry Type	V,K	K0-5; See DL205 V-memory map - Data Words
Pulse Time	V,K	K0-65535; See DL205 V-memory map - Data Words
Preset Count	V,K	K0-2147434528; See DL205 V-memory map
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

CTRADPT Example

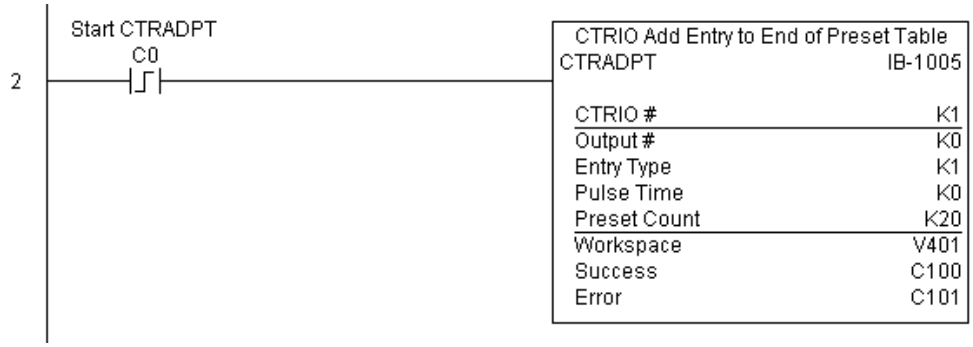
Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.

NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

Rung 2: This rung is a sample method for enabling the CTRADPT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTRADPT instruction to add a new preset to the preset table for output #0 on the CTRIO in slot 2. The new preset will be a command to RESET (entry type K1=reset), pulse time is left at zero as the reset type does not use this, and the count at which it will reset will be 20.

Operating procedure for this example code is to load the CTRADPT_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on and stay on for all counts past 10. Now reset the counter with C1, enable C0 to execute CTRADPT command to add a reset for output #0 at a count of 20, turn on C2 to enable output #0, then turn encoder to value of 10+ (output #0 should turn on) and then continue on to count of 20+ (output #0 should turn off).



(Example continued on next page)

CTRADPT Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.

Rung 4: This rung allows the operator to enable output #0 from the ladder code.

CTRIO Clear Preset Table (CTRCLRT) (IB-1007)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

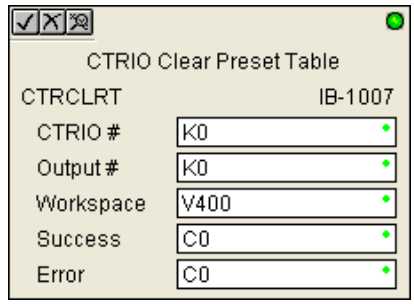
CTRIO Clear Preset Table will clear the RAM-based Preset Table on a leading edge transition to this IBox. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

DS5	Used
HPP	N/A

CTRCLRT Parameters

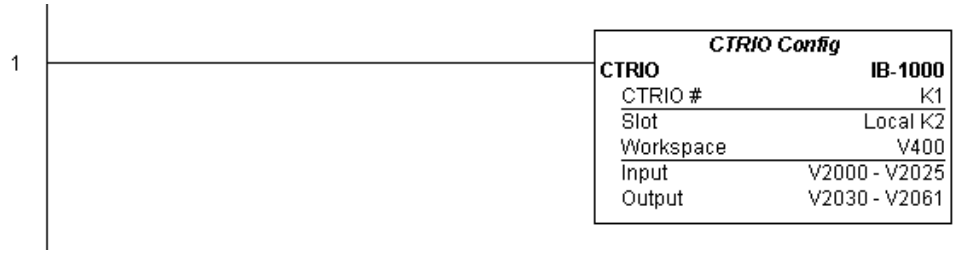
- CTRIO#: specifies a specific CTRIO module based on a user-defined number (see CTRIO Config)
- Output#: specifies a CTRIO output to be used by the instruction
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully



Parameter		DL205 Range
CTRIO#	K	K0-255
Output#	K	K0-3
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

CTRCLRT Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



***NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

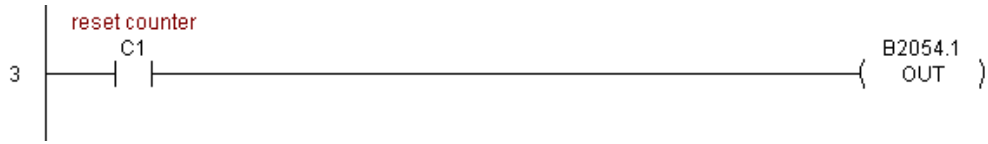
Rung 2: This rung is a sample method for enabling the CTRCLRT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTRCLRT instruction to clear the preset table for output #0 on the CTRIO in slot 2.

Operating procedure for this example code is to load the CTRCLRT_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on and stay on until a count of 20 is reached, where it will turn off. Now reset the counter with C1, enable C0 to execute CTRCLRT command to clear the preset table, turn on C2 to enable output #0, then turn encoder to value of 10+ (output #0 should NOT turn on).

CTRCLRT Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.

CTRIO Edit Preset Table Entry (CTREDPT) (IB-1003)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

CTRIO Edit Preset Table Entry, on a leading edge transition to this IBox, will edit a single entry in a Preset Table on a specific CTRIO Output resource. This IBox is good if you are editing more than one entry in a file at a time. If you wish to do just one edit and then reload the table immediately, see the CTRIO Edit and Reload Preset Table Entry (CTREDRL) IBox.

This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

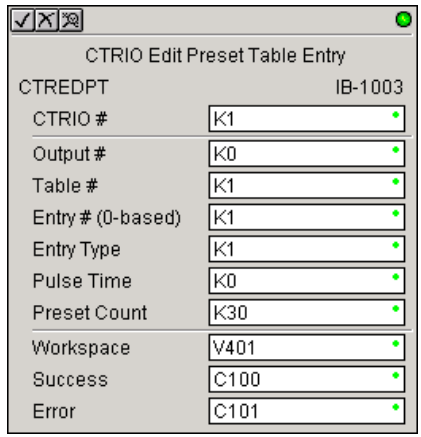
K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

CTREDPT Parameters

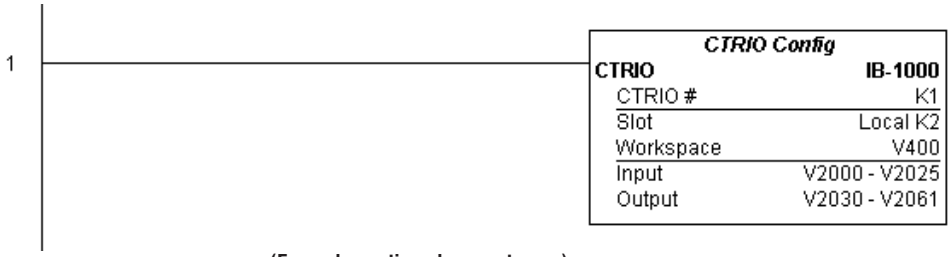


- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Table#: specifies the Table number of which an Entry is to be edited
- Entry#: specifies the Entry location in the Preset Table to be edited
- Entry Type: specifies the Entry Type to add during the edit
- Pulse Time: specifies a pulse time in msecs for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter		DL205 Range
CTRIO#	K	K0-255
Output#	K	K0-3
Table#	V,K	K0-255; See DL205 V-memory map - Data Words
Entry#	V,K	K0-255; See DL205 V-memory map - Data Words
Entry Type	V,K	K0-5; See DL205 V-memory map - Data Words
Pulse Time	V,K	K0-65535; See DL205 V-memory map - Data Words
Preset Count	V,K	K0-2147434528; See DL205 V-memory map
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

CTREDPT Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



(Example continued on next page)

***NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

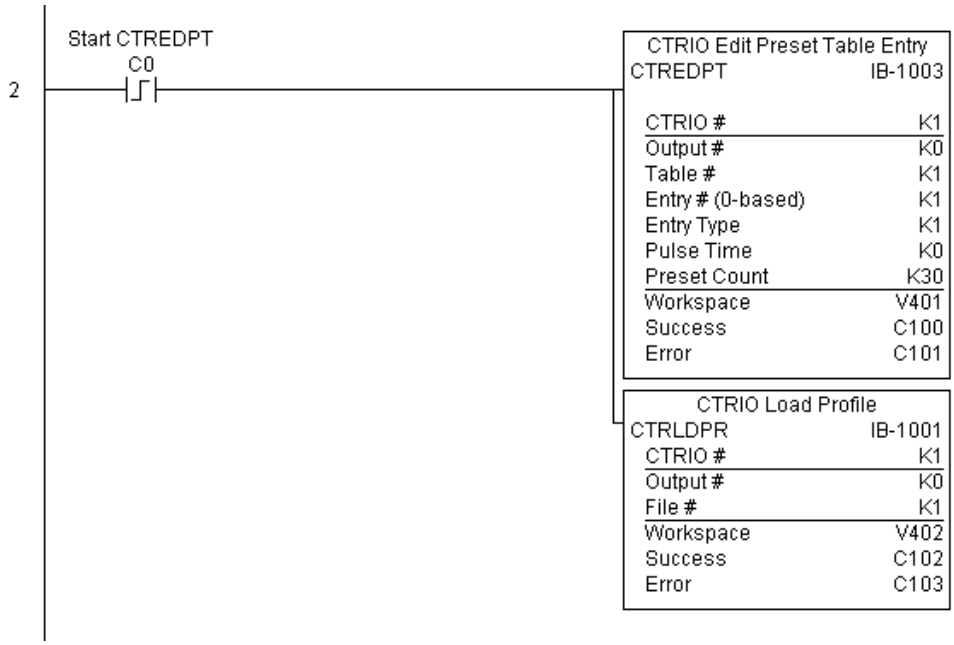
CTREDPT Example (cont'd)

Rung 2: This rung is a sample method for enabling the CTREDPT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTREDPT instruction to change the second preset from a reset at a count of 20 to a reset at a count of 30 for output #0 on the CTRIO in slot 2.

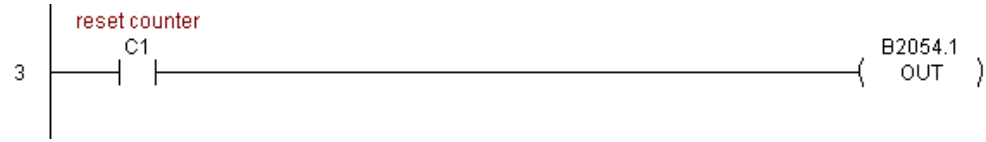
Operating procedure for this example code is to load the CTREDPT_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on and stay on until a count of 20 is reached, where it will turn off. Now reset the counter with C1, enable C0 to execute CTREDPT command to change the second preset, turn on C2 to enable output #0, then turn encoder to value of 10+ (output #0 should turn on) and then continue past a count of 30 (output #0 should turn off).

Note that we must also reload the profile after changing the preset(s); this is why the CTRLDPR command follows the CTREDPT command in this example.



CTREDPT Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.

CTRIO Edit Preset Table Entry and Reload (CTREDRL) (IB-1002)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

CTRIO Edit Preset Table Entry and Reload, on a leading edge transition to this IBox, will perform this dual operation to a CTRIO Output resource in one CTRIO command. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

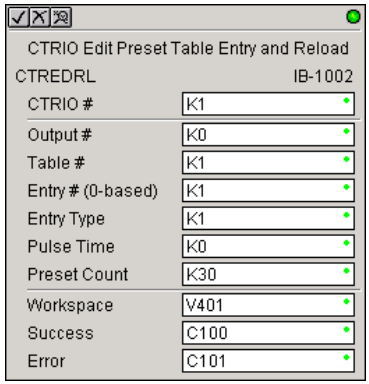
K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

CTREDRL Parameters

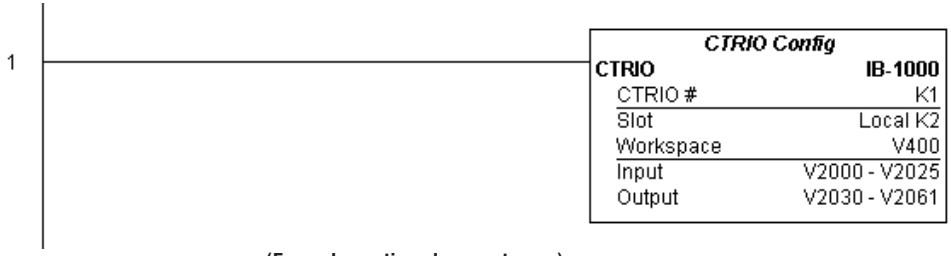


- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Table#: specifies the Table number of which an Entry is to be edited
- Entry#: specifies the Entry location in the Preset Table to be edited
- Entry Type: specifies the Entry Type to add during the edit
- Pulse Time: specifies a pulse time in msecs for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter		DL205 Range
CTRIO#	K	K0-255
Output#	K	K0-3
Table#	V,K	K0-255; See DL205 V-memory map - Data Words
Entry#	V,K	K0-255; See DL205 V-memory map - Data Words
Entry Type	V,K	K0-5; See DL205 V-memory map - Data Words
Pulse Time	V,K	K0-65535; See DL205 V-memory map - Data Words
Preset Count	V,K	K0-2147434528; See DL205 V-memory map
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

CTREDRL Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



(Example continued on next page)

***NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

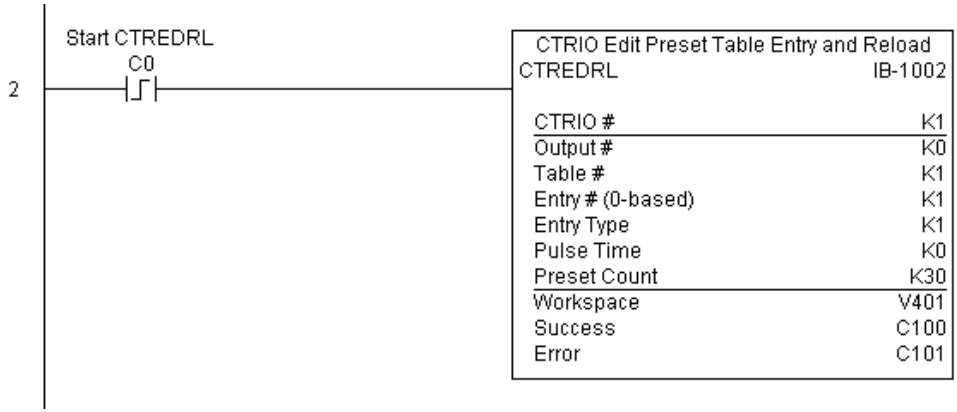
CTREDRL Example (cont'd)

Rung 2: This rung is a sample method for enabling the CTREDRL command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTREDRL instruction to change the second preset in file 1 from a reset value of 20 to a reset value of 30.

Operating procedure for this example code is to load the CTREDRL_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on, continue to a count above 20 and the output #0 light will turn off. Now reset the counter with C1, enable C0 to execute CTREDRL command to change the second preset count value to 30, then turn encoder to value of 10+ (output #0 should turn on) and continue on to a value of 30+ and the output #0 light will turn off.

Note that it is not necessary to reload this file separately, however, the command can only change one value at a time.



(Example continued on next page)

CTREDRL Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.

Rung 4: This rung allows the operator to enable output #0 from the ladder code.

CTRIO Initialize Preset Table (CTRINPT) (IB-1004)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

CTRIO Initialize Preset Table, on a leading edge transition to this IBox, will create a single entry Preset Table in memory but not as a file, on a specific CTRIO Output resource. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

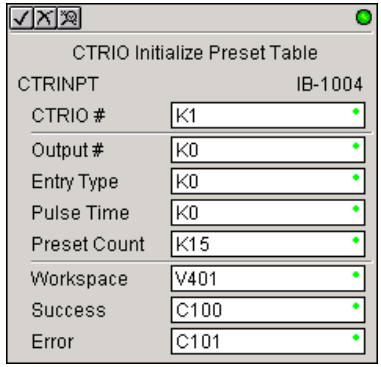
K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

CTRINPT Parameters

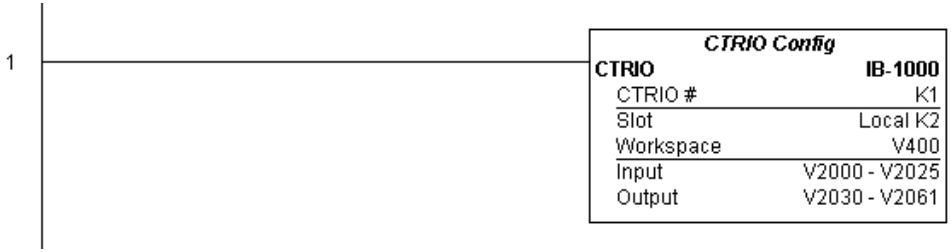


- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Entry Type: specifies the Entry Type to add during the edit
- Pulse Time: specifies a pulse time in msecs for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter		DL205 Range
CTRIO#	K	K0-255
Output#	K	K0-3
Entry Type	V,K	K0-5; See DL205 V-memory map - Data Words
Pulse Time	V,K	K0-65535; See DL205 V-memory map - Data Words
Preset Count	V,K	K0-2147434528; See DL205 V-memory map
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

CTRINPT Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



(Example continued on next page)

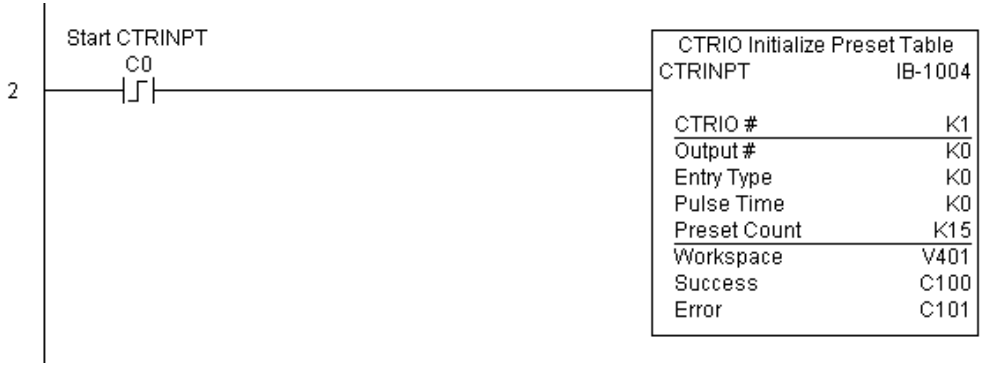
***NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

CTRINPT Example (cont'd)

Rung 2: This rung is a sample method for enabling the CTRINPT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTRINPT instruction to create a single entry preset table, but not as a file, and use it for the output #0. In this case the single preset will be set at a count of 15 for output #0.

Operating procedure for this example code is to load the CTRINPT_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 15 and output #0 light will not come on. Now reset the counter with C1, enable C0 to execute CTRINPT command to create a single preset table with a preset to set output#0 at a count of 15, then turn encoder to value of 15+ (output #0 should turn on).



CTRINPT Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.

Rung 4: This rung allows the operator to enable output #0 from the ladder code.

CTRIO Initialize Preset Table on Reset (CTRINTR) (IB-1010)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

CTRIO Initialize Preset Table on Reset, on a leading edge transition to this IBox, will create a single entry Preset Table in memory but not as a file, on a specific CTRIO Output resource. This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

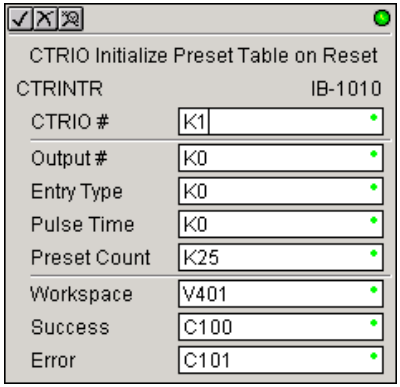
K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

CTRINTR Parameters

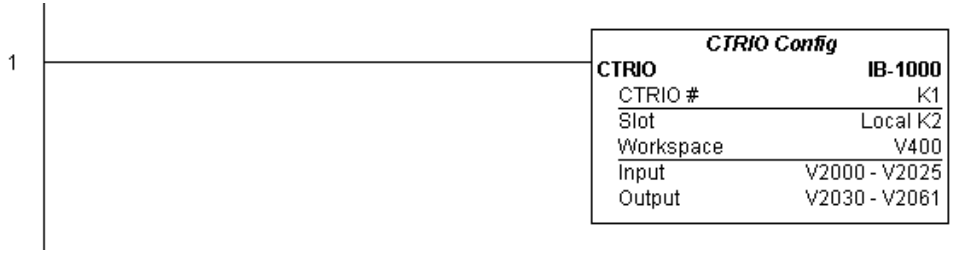


- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Entry Type: specifies the Entry Type to add during the edit
- Pulse Time: specifies a pulse time in msecs for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter		DL205 Range
CTRIO#	K	K0-255
Output#	K	K0-3
Entry Type	V,K	K0-5; See DL205 V-memory map - Data Words
Pulse Time	V,K	K0-65535; See DL205 V-memory map - Data Words
Preset Count	V,K	K0-2147434528; See DL205 V-memory map
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

CTRINTR Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



(Example continued on next page)

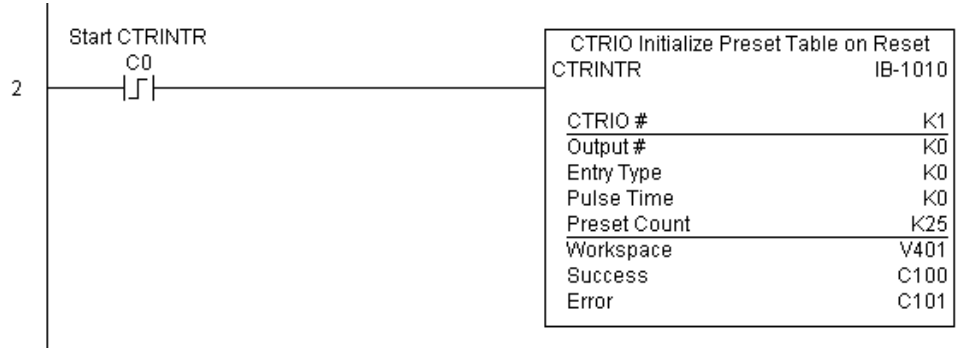
NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

CTRINTR Example (cont'd)

Rung 2: This rung is a sample method for enabling the CTRINTR command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

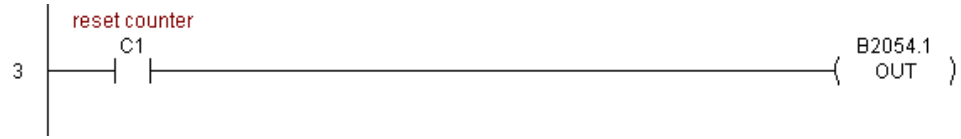
Turning on C0 will cause the CTRINTR instruction to create a single entry preset table, but not as a file, and use it for output #0, the new preset will be loaded when the current count is reset. In this case the single preset will be a set at a count of 25 for output #0.

Operating procedure for this example code is to load the CTRINTR_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on. Now turn on C0 to execute the CTRINTR command, reset the counter with C1, then turn encoder to value of 25+ (output #0 should turn on).



CTRINTR Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.

CTRIO Load Profile (CTRLDPR) (IB-1001)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

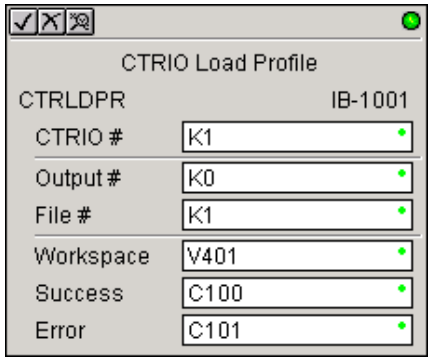
CTRIO Load Profile loads a CTRIO Profile File to a CTRIO Output resource on a leading edge transition to this IBox. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

DS5	Used
HPP	N/A

CTRLDPR Parameters

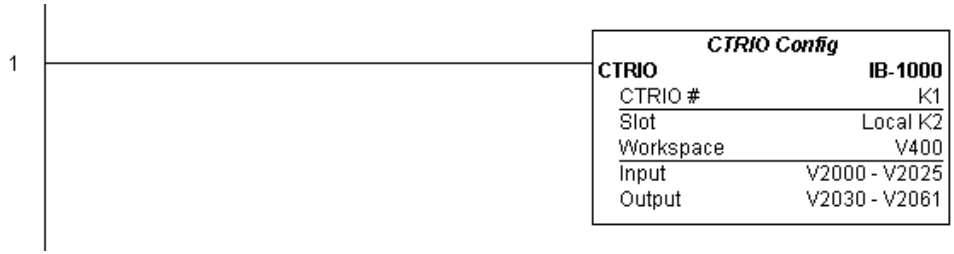
- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config)
- Output#: specifies a CTRIO output to be used by the instruction
- File#: specifies a CTRIO profile File number to be loaded
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully



Parameter		DL205 Range
CTRIO#	K	K0-255
Output#	K	K0-3
File#	V,K	K0-255; See DL205 V-memory map - Data Words
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

CTRLDPR Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



***NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: This CTRIO Load Profile IBox will load File #1 into the working memory of Output 0 in CTRIO #1. This example program requires that you load CTRLDPR_IBox.cwb into your Hx-CTRIO(2) module.

(Example continued on next page)

CTRLDPR Example (cont'd)

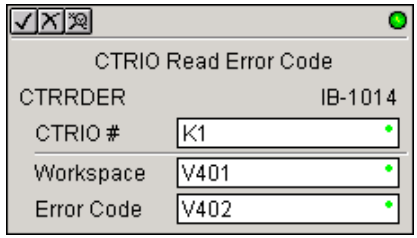
Rung 3: If the file is loaded successfully, set Profile_Loaded.

CTRIO Read Error (CTRRDER) (IB-1014)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

CTRIO Read Error Code, on a leading edge transition to this IBox, will read the decimal error code value (listed below) from the CTRIO module and place it in the specified Error Code register. This instruction is not supported when the CTRIO is used in an ERM/EBC configuration.



Since the Error Code in the CTRIO is only maintained until another CTRIO command is given, you must use this instruction immediately after the CTRIO IBox that reports an error via its Error bit parameter.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

Error Codes:

0: No Error

100: Specified command code is unknown or unsupported

101: File number not found in the file system

102: File type is incorrect for specified output function

103: Profile type is unknown

104: Specified input is not configured as a limit on this output

105: Specified limit input edge is out of range

106: Specified input function is unconfigured or invalid

107: Specified input function number is out of range

108: Specified preset function is invalid

109: Preset table is full

110: Specified Table entry is out of range

111: Specified register number is out of range

112: Specified register is an unconfigured input or output

2001: Error reading Error Code - cannot access CTRIO via ERM

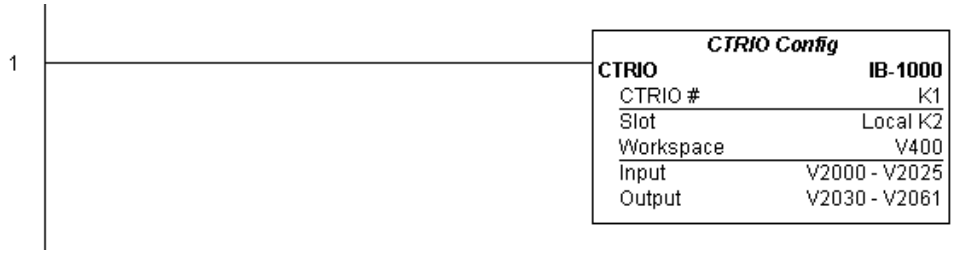
CTRRDER Parameters

- CTRIO#: specifies a specific CTRIO module based on a user-defined number (see CTRIO Config)
- Workspace: specifies a V-memory location that will be used by the instruction
- Error Code: specifies the location where the Error Code will be written

Parameter		DL205 Range
CTRIO#	K	K0-255
Workspace	V	See DL205 V-memory map - Data Words
Error Code	V	See DL205 V-memory map - Data Words

CTRRDER Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



***NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: This CTRIO Read Error Code IBox will read the Extended Error information from CTRIO #1. This example program requires that you load CTRRDER_IBox.cwb into your Hx-CTRIO(2) module.

CTRIO Run to Limit Mode (CTRRTLM) (IB-1011)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

CTRIO Run To Limit Mode, on a leading edge transition to this IBox, loads the Run to Limit command and given parameters on a specific Output resource. The CTRIO's Input(s) must be configured as Limit(s) for this function to work.

Valid Hexadecimal Limit Values:

K00 - Rising Edge of Ch1/C

K10 - Falling Edge of Ch1/C

K20 - Both Edges of Ch1/C

K01 - Rising Edge of Ch1/D

K11 - Falling Edge of Ch1/D

K21 - Both Edges of Ch1/D

K02 - Rising Edge of Ch2/C

K12 - Falling Edge of Ch2/C

K22 - Both Edges of Ch2/C

K03 - Rising Edge of Ch2/D

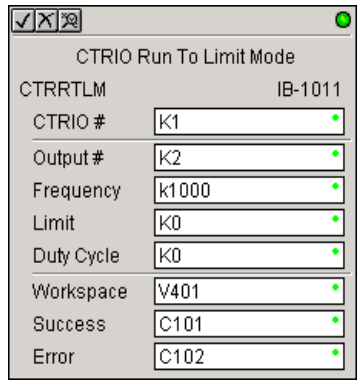
K13 - Falling Edge of Ch2/D

K23 - Both Edges of Ch2/D

This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

CTRRTLM Parameters

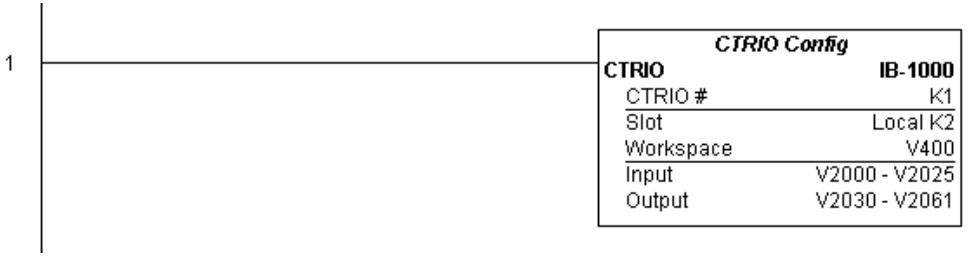


- CTRIO#: specifies a specific CTRIO module based on a user-defined number (see CTRIO Config Ibox).
- Output#: specifies a CTRIO output to be used by the instruction.
- Frequency: specifies the output pulse rate (H2-CTRIO: 20Hz - 25KHz / H2-CTRIO2: 20Hz - 250 KHz).
- Limit: the CTRIO's Input(s) must be configured as Limit(s) for this function to operate.
- Duty Cycle: specifies the % of on time versus off time. This is a hex number. Default of 0 is 50%, also entering 50 will yield 50%. 50% duty cycle is defined as on half the time and off half the time.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the instruction has completed successfully.
- Error: specifies a bit that will turn on if the instruction does not complete successfully.

Parameter		DL205 Range
CTRIO#	K	K0-255
Output#	K	K0-3
Frequency	V,K	K20-20000; See DL205 V-memory map - Data Words
Limit	V,K	K0-FF; See DL205 V-memory map - Data Words
Duty Cycle	V,K	K0-99; See DL205 V-memory map - Data Words
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

CTRRTLM Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

Rung 2: This CTRIO Run To Limit Mode IBox sets up Output #2 in CTRIO #1 to output pulses at a Frequency of 1000 Hz until Limit #0 comes on. This example program requires that you load CTRRTLM_IBox.cwb into your Hx-CTRIO(2) module.

CTRRTLM Example (cont'd)

Rung 3: If the Run To Limit Mode parameters are OK, set the Direction Bit and Enable the output.

CTRIO Run to Position Mode (CTRRTPM) (IB-1012)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

DS5	Used
HPP	N/A

CTRIO Run To Position Mode, on a leading edge transition to this IBox, loads the Run to Position command and given parameters on a specific Output resource.

Valid Function Values are:

00: Less Than Ch1/Fn1

10: Greater Than Ch1/Fn1

01: Less Than Ch1/Fn2

11: Greater Than Ch1/Fn2

02: Less Than Ch2/Fn1

12: Greater Than Ch2/Fn1

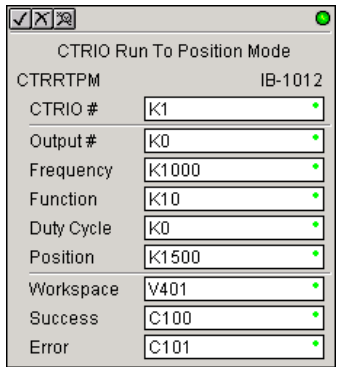
03: Less Than Ch2/Fn2

13: Greater Than Ch2/Fn2

This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

CTRRTPM Parameters

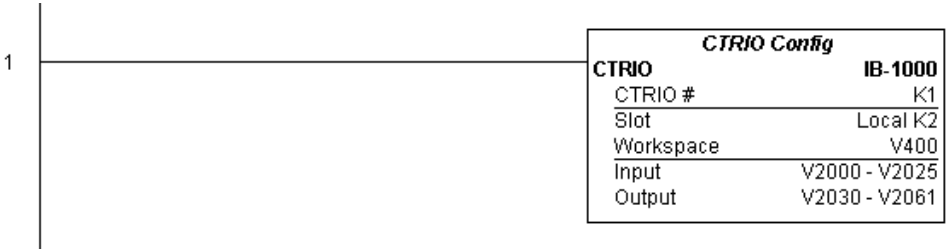


- CTRIO#: specifies a specific CTRIO module based on a user-defined number (see CTRIO Config Ibox).
- Output#: specifies a CTRIO output to be used by the instruction.
- Frequency: specifies the output pulse rate (H2-CTRIO: 20Hz - 25KHz / H2-CTRIO2: 20Hz - 250 KHz).
- Duty Cycle: specifies the % of on time versus off time. This is a hex number. Default of 0 is 50%, also entering 50 will yield 50%. 50% duty cycle is defined as on half the time and off half the time.
- Position: specifies the count value, as measured on the encoder input, at which the output pulse train will be turned off.
- Workspace: specifies a V-memory location that will be used by the instruction.
- Success: specifies a bit that will turn on once the instruction has completed successfully.
- Error: specifies a bit that will turn on if the instruction does not complete successfully.

Parameter		DL205 Range
CTRIO#	K	K0-255
Output#	K	K0-3
Frequency	V,K	K20-20000; See DL205 V-memory map - Data Words
Duty Cycle	V,K	K0-99; See DL205 V-memory map
Position	V,K	K0-2147434528; See DL205 V-memory map
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

CTRRTPM Example

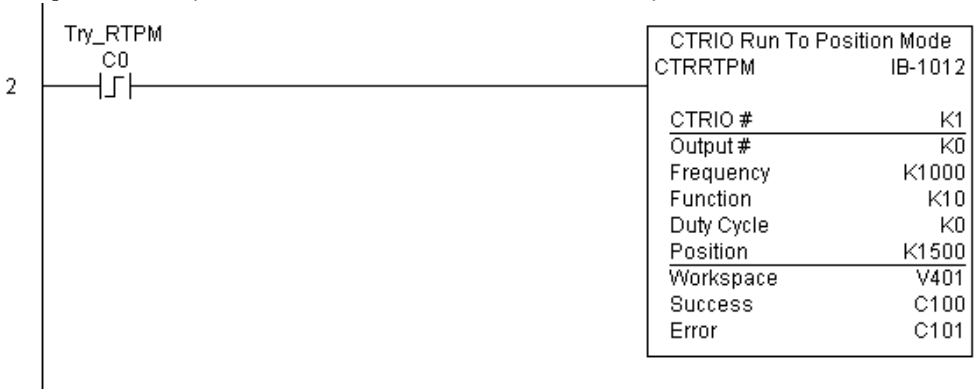
Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters.

CTRRTPM Example (cont'd)

Rung 2: This CTRIO Run To Position Mode IBox sets up Output #0 in CTRIO #1 to output pulses at a Frequency of 1000 Hz, use the 'Greater than Ch1/Fn1' comparison operator, until the input position of 1500 is reached. This example program requires that you load CTRRTPM_IBox.cwb into your Hx-CTRIO(2) module.



Rung 3: If the Run To Position Mode parameters are OK, set the Direction Bit and Enable the output.

CTRIO Velocity Mode (CTRVELO) (IB-1013)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

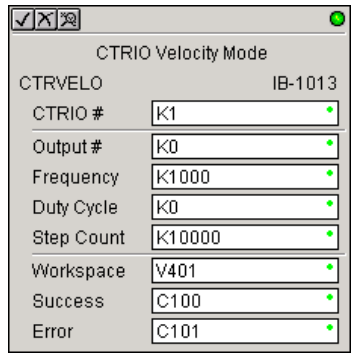
CTRIO Velocity Mode loads the Velocity command and given parameters on a specific Output resource on a leading edge transition to this IBox.

This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

DS5	Used
HPP	N/A

CTRVELO Parameters

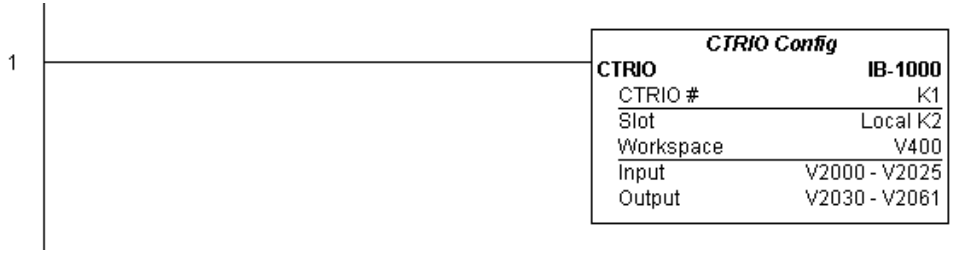


- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Frequency: specifies the output pulse rate (H2-CTRIO: 20Hz - 25KHz / H2-CTRIO2: 20Hz - 250 KHz)
- Duty Cycle: specifies the % of on time versus off time. This is a hex number. Default of 0 is 50%, also entering 50 will yield 50%. 50% duty cycle is defined as on half the time and off half the time
- Step Count: This DWORD value specifies the number of pulses to output. A Step Count value of -1 (or 0xFFFFFFFF) causes the CTRIO to output pulses continuously. Negative Step Count values must be V-Memory references.
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter		DL205 Range
CTRIO#	K	K0-255
Output#	K	K0-3
Frequency	V,K	K20-20000; See DL205 V-memory map - Data Words
Duty Cycle	V,K	K0-99; See DL205 V-memory map
Step Count	V,K	K0-2147434528; See DL205 V-memory map
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

CTRVELO Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



***NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: This CTRIO Velocity Mode IBox sets up Output #0 in CTRIO #1 to output 10,000 pulses at a Frequency of 1000 Hz. This example program requires that you load CTRVELO_IBox.cwb into your Hx-CTRIO(2) module.

CTRVELO Example (cont'd)

Rung 3: If the Velocity Mode parameters are OK, set the Direction Bit and Enable the output.

CTRIO Write File to ROM (CTRWFTR) (IB-1006)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

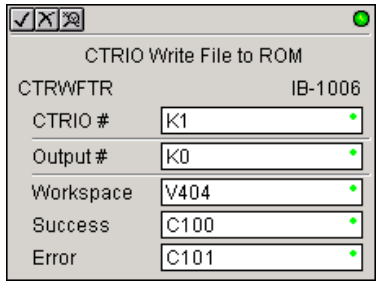
CTRIO Write File to ROM writes the runtime changes made to a loaded CTRIO Preset Table back to Flash ROM on a leading edge transition to this IBox. This IBox will take more than one PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

DS5	Used
HPP	N/A

CTRWFTR Parameters

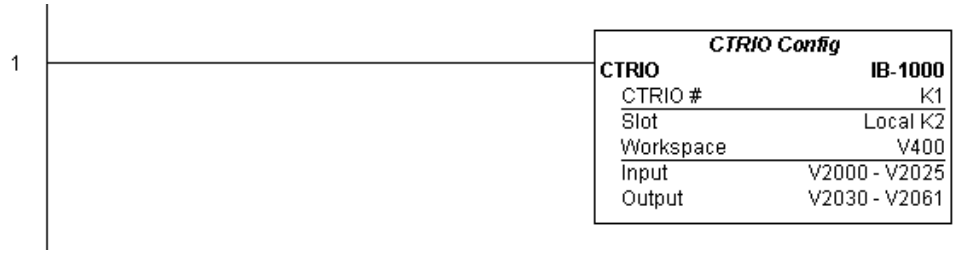
- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has completed successfully
- Error: specifies a bit that will turn on if the instruction does not complete successfully



Parameter		DL205 Range
CTRIO#	K	K0-255
Output#	K	K0-3
Workspace	V	See DL205 V-memory map - Data Words
Success	X,Y,C,GX,GY,B	See DL205 V-memory map
Error	X,Y,C,GX,GY,B	See DL205 V-memory map

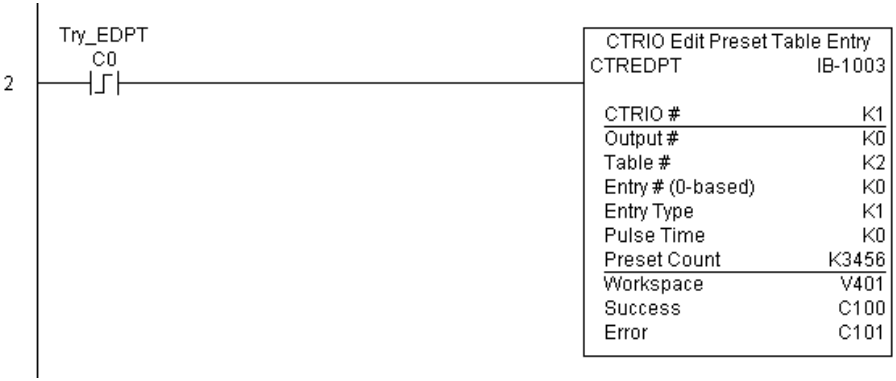
CTRWFTR Example

Rung 1: This sets up the CTRIO module in slot 2 of the local base. Each CTRIO module in the system will need a separate CTRIO Config IBox before any CTRxxxx IBoxes can be used. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



***NOTE**: The CTRIO Configuration IBox instruction does not require a permissive contact. The top line will be identified in **BOLD italics**, and the instruction name and ID will be in **BOLD characters**.*

Rung 2: This CTRIO Edit Preset Table Entry IBox will change Entry 0 in Table #2 to be a RESET at Count 3456. This example program requires that you load CTRWFTR_IBox.cwb into your Hx-CTRIO(2) module.



(Example continued on next page)

CTRWFTR Example (cont'd)

Rung 3: If the file is successfully edited, use a Write File To ROM IBox to save the edited table back to the CTRIO's ROM, thereby making the changes retentive.

Drum Instruction Programming (D2-250-1, D2-260 and D2-262 Only)

In This Chapter...

Introduction

Purpose

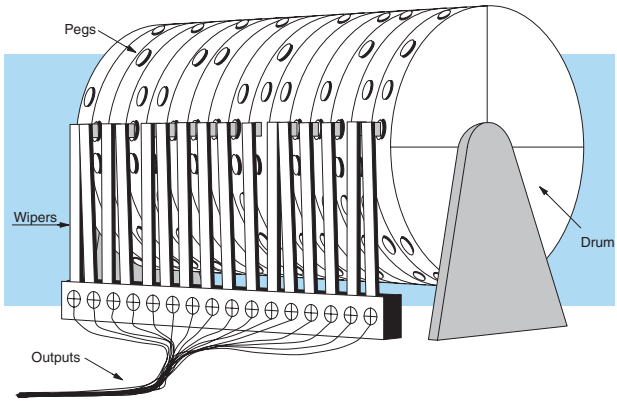
The four types of drum instructions available in the D2-250-1, D2-260 and D2-262 CPUs electronically simulate an electro-mechanical drum sequencer. The instructions offer slight variations on the basic principle.

Drum Terminology

Drum instructions are best suited for repetitive processes that consist of a finite number of steps. They can do the work of many rungs of ladder logic with elegant simplicity. Therefore, drums can save a lot of programming and debugging time.

We introduce some terminology associated with the **drum** instruction by describing the original mechanical drum shown below. The mechanical drum generally has pegs on its curved surface. The pegs are populated in a particular **pattern**, representing a set of desired actions for machine control. A motor or solenoid rotates the drum a precise amount at specific times. During rotation, stationary wipers sense the presence of pegs (present = on, absent = off). This interaction makes or breaks electrical contact with the wipers, creating electrical **outputs** from the drum. The outputs are wired to devices on a machine for On/Off control.

Drums usually have a finite number of positions within one rotation, called **steps**. Each step represents some process step. At powerup, the drum **resets** to a particular step. The drum rotates from one step to the next based on a **timer**, or on some external **event**. During special conditions, a machine operator can manually increment the drum step using a **jog** control on the drum's drive mechanism. The contact closure of each wiper generates a unique on/off pattern called a **sequence**, designed for controlling a specific machine. Because the drum is circular, it automatically repeats the sequence once per rotation. Applications vary greatly, and a particular drum may rotate once per second, or as slowly as once per week.



Electronic drums provide the benefits of mechanical drums and more. For example, they have a **preset** feature that is impossible for mechanical drums: The preset function lets you move from the present step directly to any other step on command!

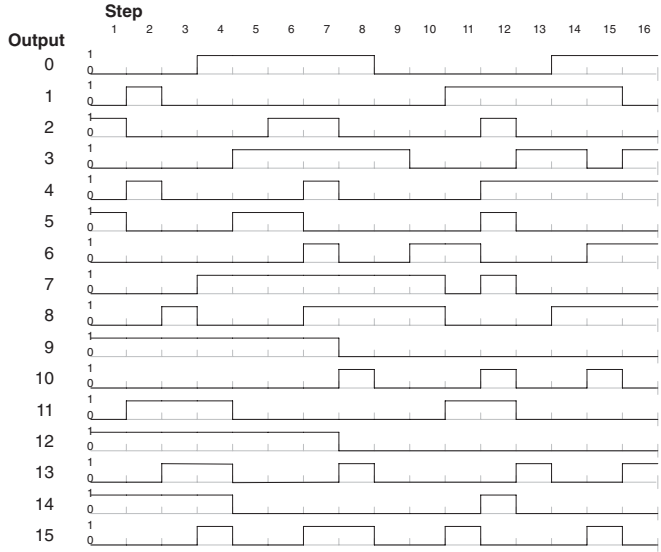
Drum Chart Representation

For editing purposes, the electronic drum is presented in chart form in *Direct*SOFT and in this manual. Imagine slicing the surface of a hollow drum cylinder between two rows of pegs, then pressing it flat. Now you can view the drum as a chart as shown below. Each row represents a step, numbered 1 through 16. Each column represents an output, numbered 0 through 15 (to match word bit numbering). The solid circles in the chart represent pegs (On state) in the mechanical drum, and the open circles are empty peg sites (Off state).

	OUTPUTS															
STEP	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	○	●	○	●	○	○	●	○	○	○	●	○	○	●	○	○
2	○	●	○	●	●	○	●	○	○	○	○	●	○	○	●	○
3	○	●	●	●	●	○	●	●	○	○	○	○	○	○	○	○
4	●	●	●	●	●	○	●	○	●	○	○	○	○	○	○	●
5	○	○	○	●	○	○	●	○	●	○	●	○	●	○	○	●
6	○	○	○	●	○	○	●	○	●	○	●	○	●	●	○	●
7	●	●	○	●	○	○	●	●	●	●	○	●	●	●	○	●
8	●	○	●	○	○	●	○	●	●	○	○	○	●	○	○	●
9	○	○	○	○	○	○	○	●	●	○	○	○	●	○	○	○
10	○	○	○	○	○	○	○	●	●	●	○	○	○	○	○	○
11	●	○	○	○	●	○	○	○	○	●	○	○	○	○	●	○
12	○	●	○	○	●	●	○	○	●	○	●	●	○	●	●	○
13	○	○	●	○	○	○	○	○	○	○	○	●	●	○	●	○
14	○	○	○	○	○	○	○	●	○	○	○	●	●	○	●	●
15	●	○	○	○	○	●	○	●	○	●	○	●	○	○	●	●
16	○	○	●	○	○	○	○	●	○	●	○	●	●	○	○	●

Output Sequences

The mechanical drum sequencer derives its name from sequences of control changes on its electrical outputs. The following figure shows the sequence of On/Off controls generated by the drum pattern above. Compare the two, and you will find that they are equivalent! If you can see their equivalence, you are well on your way to understanding drum instruction operation.



Step Transitions

Drum Instruction Types

There are four types of Drum instructions in the D2-250-1, D2-260 and D2-262 CPUs:

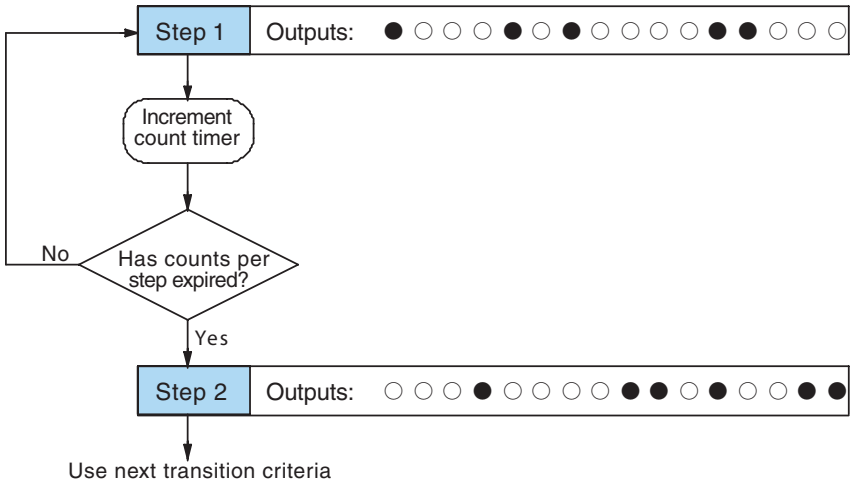
- Timed Drum with Discrete Outputs (DRUM)
- Time and Event Drum with Discrete Outputs (EDRUM)
- Masked Event Drum with Discrete Outputs (MDRMD)
- Masked Event Drum with Word Output (MDRMW)

The four drum instructions include time-based step transitions, and three include event-based transitions as well. Other options include outputs defined as a single word or as individual bits, and an output mask (individual output disable/enable).

Each drum has 16 steps, and each step has 16 outputs. Referring to the figure below, each output can be either a Y or C coil, offering programming flexibility. Step 1 has been assigned an arbitrary unique output pattern (○= Off, ● = On) as shown.

Timer-Only Transitions

Drums move from one step to another based on time and/or an external event (input). Each step has its own transition condition which you assign during the drum instruction entry. The figure below shows how timer-only transitions work.



The drum remains in Step 1 for a specific duration (user-programmable). The timebase of the timer is programmable, from 0.01 seconds to 99.99 seconds. This establishes the resolution, or the duration, of each "tick of the clock". Each step uses the same timebase, but has its own unique counts per step, which you program. The drum spends a specific amount of time in each step, given by the formula:

Time in step = 0.01 seconds X Timebase x Counts per step

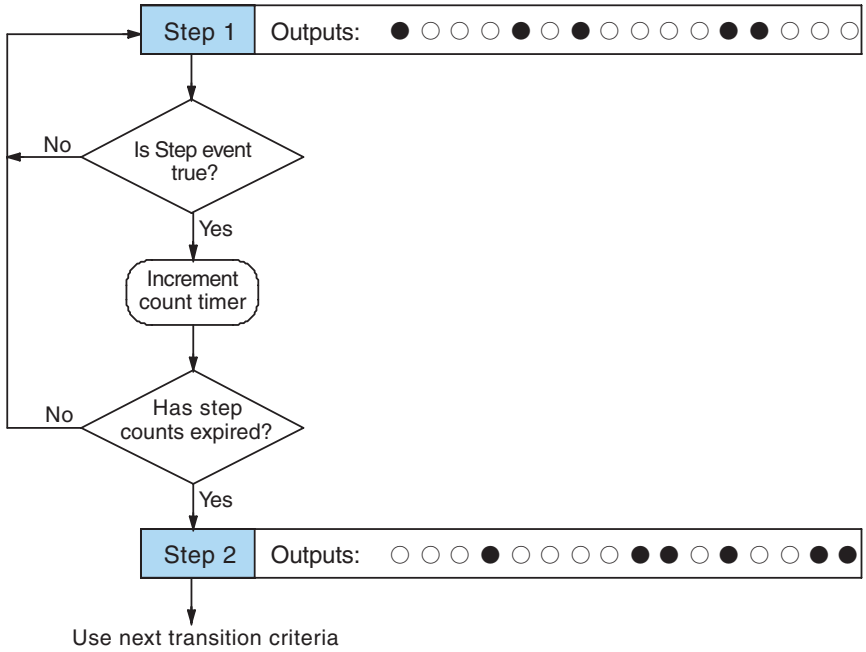
For example, if you program a 5 second time base and 12 counts for Step 1, then the drum will spend 60 seconds in Step 1. The maximum time for any step is given by the formula:

Max Time per step = 0.01 seconds X 9999 X 9999
= 999,800 seconds = 277.7 hours = 11.6 days

***NOTE**: When first choosing the timebase resolution, a good rule of thumb is to make it about 1/10 the duration of the shortest step in your drum. Then you will be able to optimize the duration of that step in 10% increments. Other steps with longer durations allow optimizing by even smaller increments (percentage-wise). Also, note that the drum instruction executes once per CPU scan. Therefore, it is pointless to specify a drum timebase that is much faster than the CPU scan time.*

Timer and Event Transitions

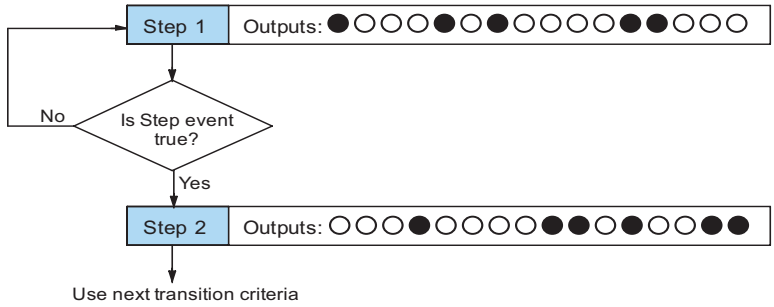
Step transitions may also occur based on time and/or external events. The figure below shows how step transitions work in these cases.



When the drum enters Step 1, it sets the output pattern as shown. Then it begins polling the external input programmed for that step. You can define event inputs as X, Y, or C discrete point types. Suppose we select X0 for the Step 1 event input. If X0 is off, then the drum remains in Step 1. When X0 is On, the event criteria is met and the timer increments. The timer increments as long as the event (X0) remains true. When the counts for Step 1 have expired, then the drum moves to Step 2. The outputs change immediately to match the new pattern for Step 2.

Event-Only Transitions

Step transitions do not require both the event and the timer criteria programmed for each step. You have the option of programming just one of the two, and even mixing transition types among all the steps of the drum. For example, you might want Step 1 to transition on an event, Step 2 to transition on time only, and Step 3 to transition on both time and an event. Furthermore, you may elect to use only part of the 16 steps, and only part of the 16 outputs.



Counter Assignments

Each drum instruction uses the resources of four counters in the CPU. When programming the drum instruction, you select the first counter number. The drum also uses the next three counters automatically. The counter bit associated with the first counter turns on when the drum has completed its cycle, going off when the drum is reset. These counter values and the counter bit precisely indicate the progress of the drum instruction, and can be monitored by your ladder program.

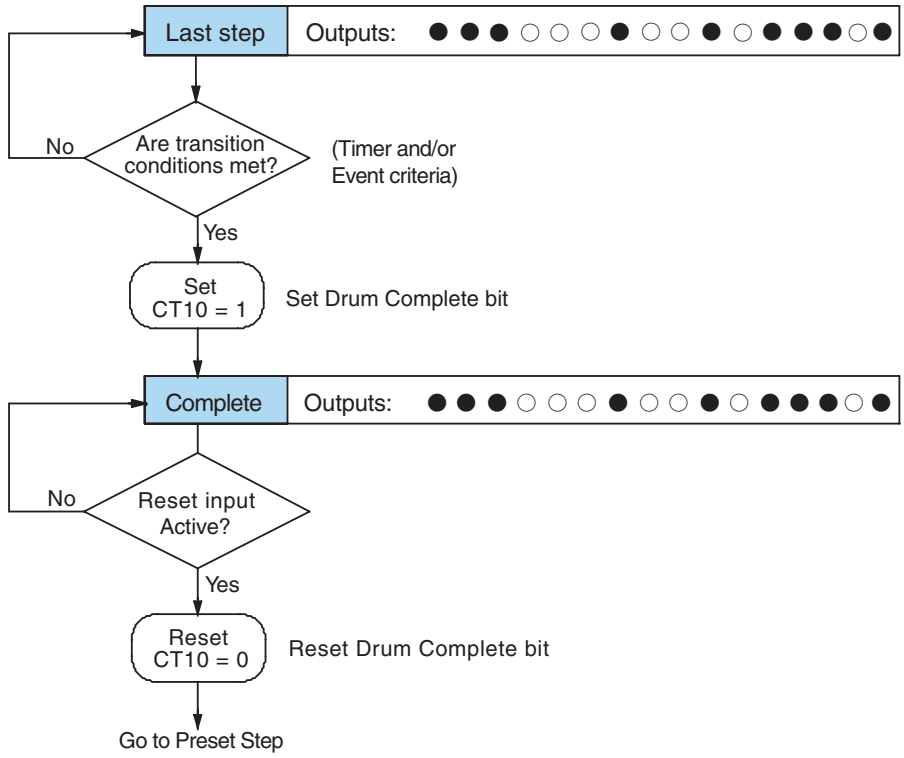
Suppose we program a timer drum to have 8 steps, and we select CT10 for the counter number (remember, counter numbering is in octal). Counter usage is shown to the right. The right column holds typical values, interpreted below.

Counter Assignments			
CT10	Counts in step	V1010	1528
CT11	Timer Value	V1011	0200
CT12	Preset Step	V1012	0001
CT13	Current Step	V1013	0004

CT10 shows that we are at the 1528th count in the current step, which is step 4 (shown in CT13). If we have programmed step 4 to have 3000 counts, then the step is just over half completed. CT11 is the count timer, shown in units of 0.01 seconds. So, each least-significant-digit change represents 0.01 seconds. The value of 200 means that we have been in the current count (1528) for 2 seconds (0.01 x 200). Finally, CT12 holds the preset step value which was programmed into the drum instruction. When the drum's Reset input is active, it presets to step 1 in this case. The value of CT12 changes only if the ladder program writes to it, or the drum instruction is edited and the program is restarted. Counter bit CT10 turns on when the drum cycle is complete, and turns off when the drum is reset.

Last Step Completion

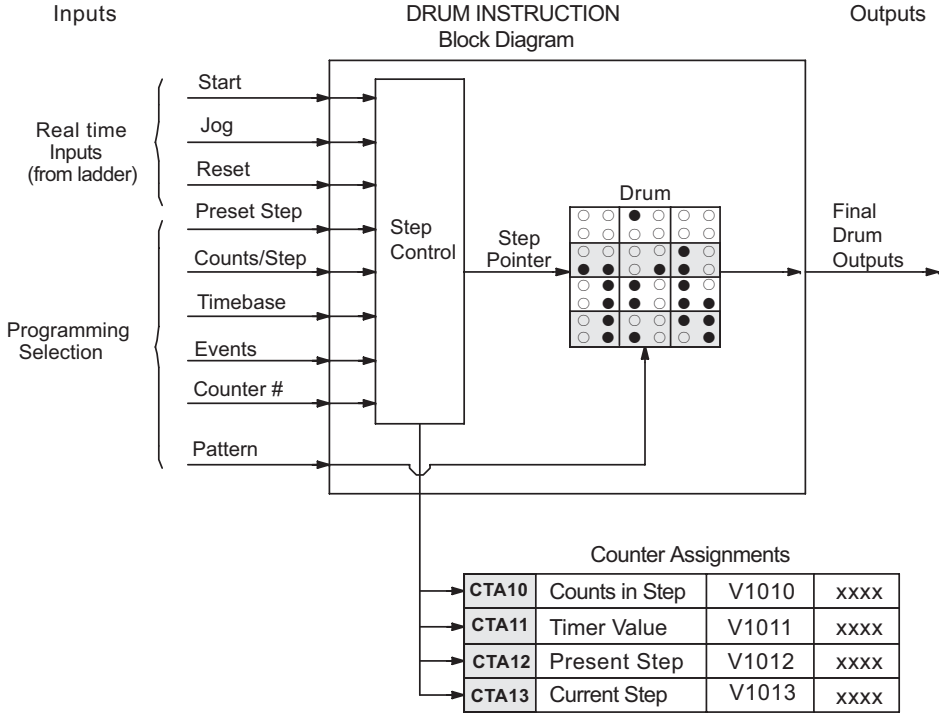
The last step in a drum sequence may be any step number, since partial drums are valid. Refer to the following figure. When the transition conditions of the last step are met, the drum sets the counter bit corresponding to the counter named in the drum instruction box (such as CT10). Then it moves to a final "drum complete" state. The drum outputs remain in the pattern defined for the last step. Having finished a drum cycle, the Start and Jog inputs have no effect at this point. The drum leaves the "drum complete" state when the Reset input becomes active (or on a program-to–run mode transition). It resets the drum complete bit (such as CT10), and then goes directly to the appropriate step number defined as the preset step.



Overview of Drum Operation

Drum Instruction Block Diagram

The drum instruction utilizes various inputs and outputs in addition to the drum pattern itself. Refer to the figure below.



The drum instruction accepts several inputs for step control, the main control of the drum. The inputs and their functions are:

- **Start** – The Start input is effective only when Reset is off. When Start is on, the drum timer runs if it is in a timed transition, and the drum looks for the input event during event transitions. When Start is off, the drum freezes in its current state (Reset must remain off), and the drum outputs maintain their current on/off pattern.
- **Jog** – The jog input is only effective when Reset is off (Start may be either on or off). The jog input increments the drum to the next step on each off-to-on transition (only EDRUM supports the jog input).
- **Reset** – The Reset input has priority over the Start input. When Reset is on, the drum moves to its preset step. When Reset is off, then the Start input operates normally.
- **Preset Step** – A step number from 1 to 16 that you define (typically is step 1). The drum moves to this step whenever Reset is on, and whenever the CPU first enters run mode.

- **Counts/Step** – The number of timer counts the drum spends in each step. Each step has its own counts parameter. However, programming the counts/step is optional.
- **Timer Value** – the current value of the counts/step timer.
- **Counter #** – The counter number specifies the first of four consecutive counters which the drum uses for step control. You can monitor these to determine the drum's progress through its control cycle.
- **Events** – Either an X, Y, C, S, T, CT, or SP type discrete point serves as step transition inputs. Each step has its own event. However, programming the event is optional on Timer/Event Drums.

WARNING: The outputs of a drum are enabled any time the CPU is in Run Mode. The Start Input does not have to be on, and the Reset input does not disable the outputs. Upon entering Run Mode, drum outputs automatically turn on or off according to the pattern of the current step of the drum. This initial step number depends on the counter memory configuration: non-retentive versus retentive.

Powerup State of Drum Registers

The choice of the starting step on powerup and program-to-run mode transitions are important to consider for your application. Please refer to the following chart. If the counter memory is configured as non-retentive, the drum is initialized the same way on every powerup or program-to-run mode transition. However, if the counter memory is configured to be retentive, the drum will stay in its previous state.

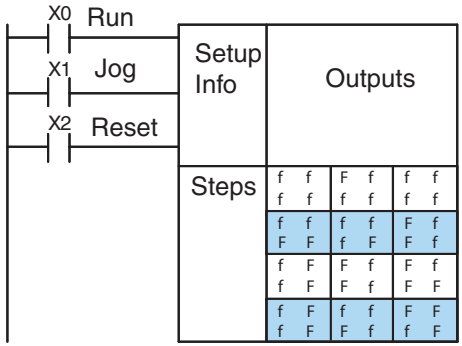
Counter Number	Function	Initialization on Powerup	
		Non-Retentive Case	Retentive Case
CTA(n)	Current Step Count	Initialize = 0	Use Previous (no change)
CTA(n + 1)	Counter Timer Value	Initialize = 0	Use Previous (no change)
CTA(n + 2)	Preset Step	Initialize = Preset Step #	Use Previous (no change)
CTA(n + 3)	Current Step #	Initialize = Preset Step #	Use Previous (no change)

Applications with relatively fast drum cycle times typically will need to be reset on powerup, using the non-retentive option. Applications with relatively long drum cycle times may need to resume at the previous point where operations stopped, using the retentive case. The default option is the retentive case. This means that if you initialize scratchpad V-memory, the memory will be retentive.

Drum Control Techniques

Drum Control Inputs

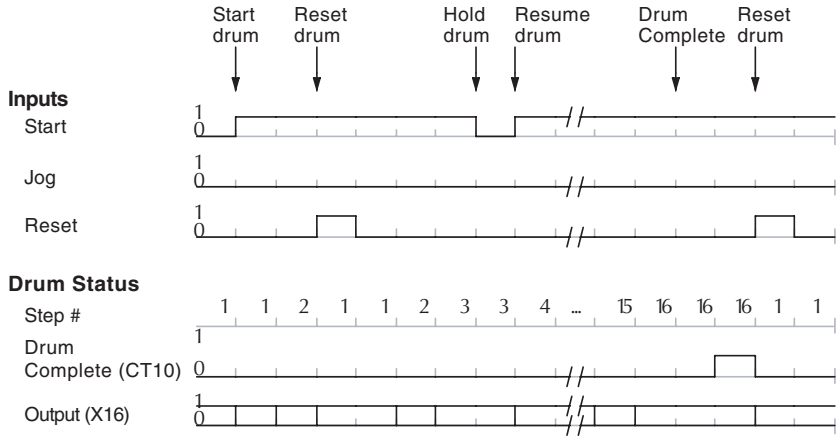
Now we are ready to put together the concepts on the previous pages and demonstrate general control of the drum instruction box. The drawing to the right shows a simplified generic drum instruction. Inputs from ladder logic control the Start, Jog, and Reset Inputs (only the EDRUM instruction supports the Jog Input). The first counter bit of the drum (CT10, for example) indicates the drum cycle is done.



The timing diagram below shows an arbitrary timer drum input sequence and how the drum responds. As the CPU enters Run mode it initializes the step number to the preset step number (typically it is Step 1). When the Start input turns on, the drum begins running, waiting for an event and/or running the timer (depends on the setup).

After the drum enters Step 2, Reset turns On while Start is still On. Since Reset has priority over Start, the drum goes to the preset step (Step 1). Note that the drum is *held* in the preset step during Reset, and that step does *not run* (respond to events or run the timer) until Reset turns off.

After the drum has entered step 3, the Start input goes off momentarily, halting the drum's timer until Start turns on again.

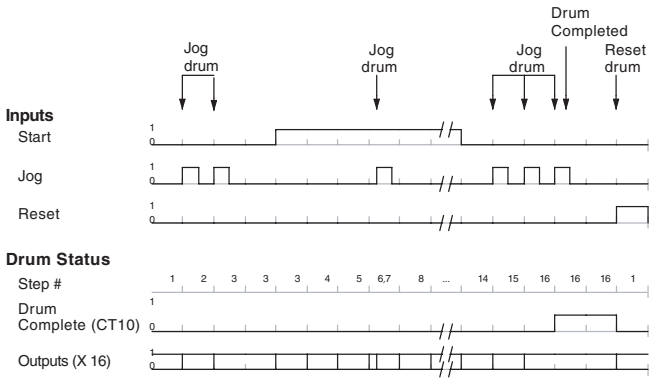


When the drum completes the last step (Step 16 in this example), the Drum Complete bit (CT10) turns on, and the step number remains at 16. When the Reset input turns on, it turns off the Drum Complete bit (CT10), and forces the drum to enter the preset step.

***NOTE**: The timing diagram shows all steps using equal time durations. Step times can vary greatly, depending on the counts/step programmed.*

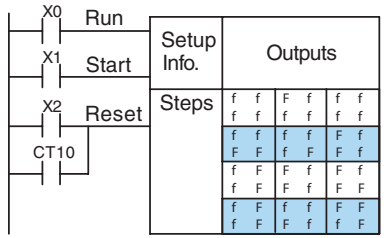
In the figure below, we focus on how the Jog input works on event drums. To the left of the diagram, note that the off-to-on transitions of the Jog input increments the step. Start may be either on or off (however, Reset must be off). Two jogs takes the drum to step 3. Next, the Start input turns on, and the drum begins running normally. During step 6 another Jog input signal occurs. This increments the drum to step 7, setting the timer to 0. The drum begins running immediately in step 7, because Start is already on. The drum advances to step 8 normally.

As the drum enters step 14, the Start input turns off. Two more Jog signals moves the drum to step 16. However, note that a third Jog signal is required to move the drum through step 16 to "drum complete." Finally, a Reset input signal arrives which forces the drum into the preset step and turns off the drum complete bit.



Self-Resetting Drum

Applications often require drums that automatically start over once they complete a cycle. This is easily accomplished using the drum complete bit. In the figure to the right, the drum instruction setup is for CT10, so we logically OR the drum complete bit (CT10) with the Reset input. When the last step is done, the drum turns on CT10 which resets itself to the preset step, also resetting CT10. Contact X2 still works as a manual reset.



Initializing Drum Outputs

The outputs of a drum are enabled any time the CPU is in run mode. On program-to-run mode transitions, the drum goes to the preset step, and the outputs energize according to the pattern of that step. If your application requires all outputs to be off at powerup, make the preset step in the drum a "reset step," with all outputs off.

Using Complex Event Step Transitions

Each event-based transition accepts only one contact reference for the event. However, this does not limit events to just one contact. Just use a control relay contact such as C0 for the step transition event. Elsewhere in ladder logic, you may use C0 as an output coil, making it dependent on many other "events" (contacts).

Drum Instructions

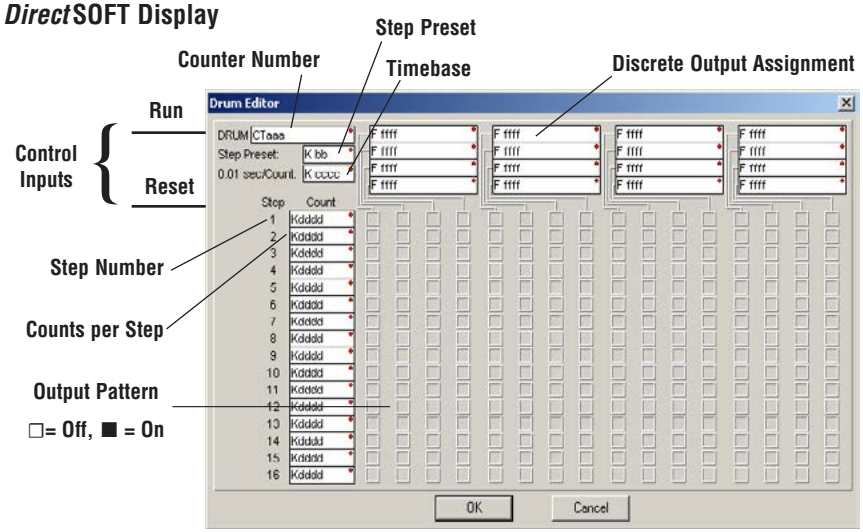
All of the D2-250-1, D2-260 and D2-262 drum instructions may be programmed using *Direct*SOFT. The EDRUM is the only drum instruction that can be programmed with a handheld programmer (firmware version v2.21 or later). This section covers entry using *Direct*SOFT for all instructions plus the handheld mnemonics for the EDRUM instruction.

Timed Drum with Discrete Outputs (DRUM)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The Timed Drum with Discrete Outputs is the most basic of the D2-250–1, D2-260 and D2-262 drum instructions. It operates according to the principles covered on the previous pages. Below is the instruction in chart form as displayed by *Direct*SOFT.

***Direct*SOFT Display**



The Timed Drum features 16 steps and 16 outputs. Step transitions occur only on a timed basis, specified in counts per step. Unused steps must be programmed with "counts per step" = 0 (this is the default entry). The discrete output points may be individually assigned as X, Y, or C types, or may be left unused. The output pattern may be edited graphically with *Direct*SOFT.

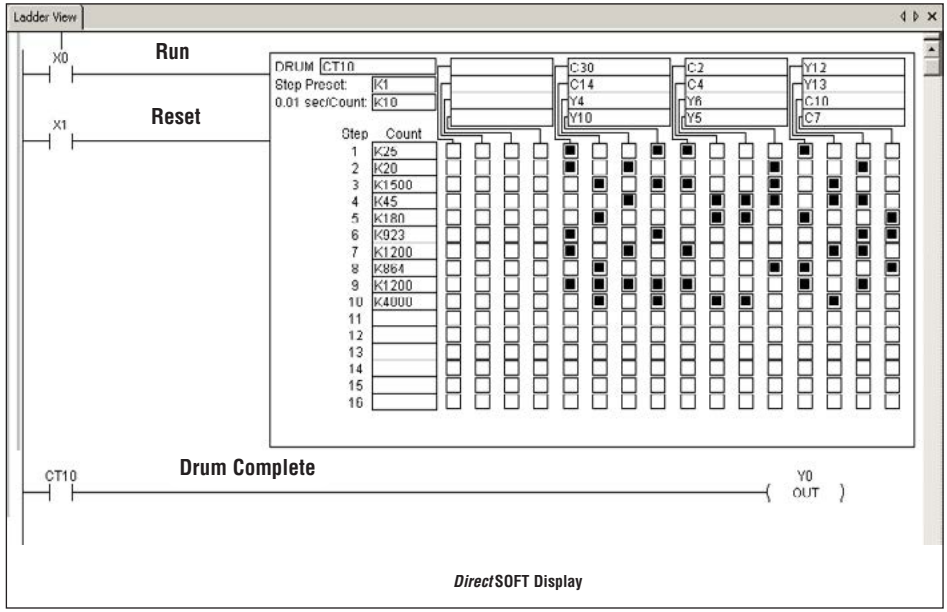
Whenever the Start input is energized, the drum's timer is enabled. It stops when the last step is complete, or when the Reset input is energized. The drum enters the preset step chosen upon a CPU program-to-run mode transition, and whenever the Reset input is energized.

Drum Parameters	Field	Data Types	Ranges
Counter Number	aaa		0–174 (D2-250-1) 0–374 (D2-260/D2-262)
Step Preset	bb	K	1–16
Timer base	cccc	K	0– 99.99 seconds
Counts per step	dddd	K	0– 9999
Discrete Outputs	F ffff	X, Y, C	see page 3-55 or page 3-56

Drum instructions use four counters in the CPU. The ladder program can read the counter values for the drum's status. The ladder program may write a new preset step number to CTA(n+2) at any time. However, the other counters are for monitoring purposes only.

Counter Number	D2-250-1 Ranges of (n)	D2-260/D2-262 Ranges of (n)	Function	Counter Bit Function
CTA(n)	0 - 174	0 - 374	Counts in step	CT(n) = Drum Complete
CTA(n+1)	1 - 175	1 - 375	Timer value	CT(n+1) = (not used)
CTA(n+2)	2 - 176	2 - 376	Preset Step	CT(n+2) = (not used)
CTA(n+3)	3 - 177	3 - 377	Current Step	CT(n+3) = (not used)

The following ladder program shows the DRUM instruction in a typical ladder program, as shown by ***Direct***SOFT. Steps 1 through 10 are used, and 12 of the 16 output points are used. The preset step is step 1. The timebase runs at (K10 x 0.01) = 0.1 second per count. Therefore, the duration of step 1 is (25 x 0.1) = 2.5 seconds. In the last rung, the Drum Complete bit (CT10) turns on output Y0 upon completion of the last step (step 10). A drum reset also resets CT10.



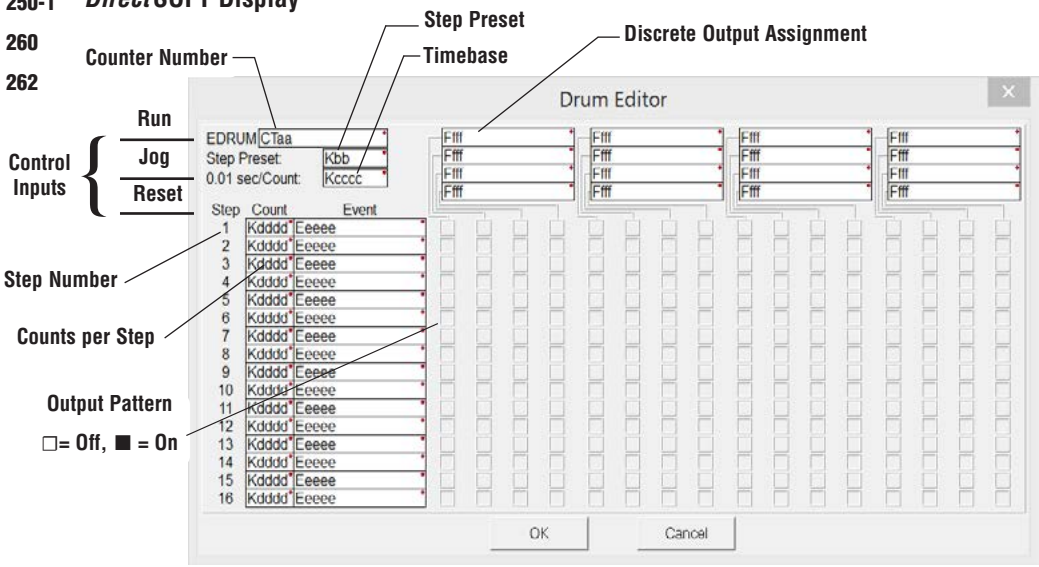
***Direct*SOFT Display**

Event Drum (EDRUM)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The Event Drum (EDRUM) features time-based and event-based step transitions. It operates according to the general principles of drum operation covered in the beginning of this chapter. Below is the instruction as displayed by *Direct*SOFT.

*Direct*SOFT Display



The Event Drum features 16 steps and 16 discrete outputs. Step transitions occur on timed and/or event basis. The jog input also advances the step on each off-to-on transition. Time is specified in counts per step, and events are specified as discrete contacts. Unused steps and events must be left blank. The discrete output points may be individually assigned.

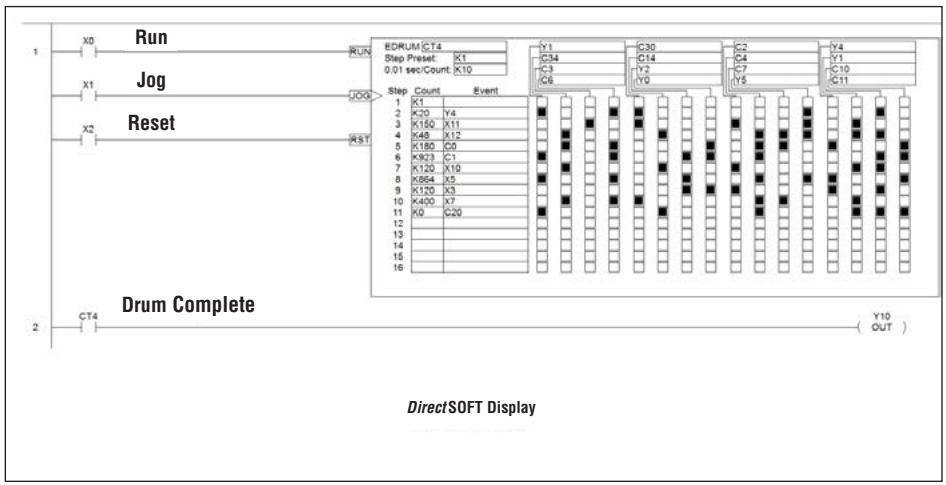
Drum Parameters	Field	Data Types	Ranges
Counter Number	aaa	-	0 - 174 (D2-250-1) 0 - 374 (D2-260/D2-262)
Step Preset	bb	K	1 - 16
Timer base	cccc	K	0 - 99.99 seconds
Counts per step	dddd	K	0 - 9999
Event	eeee	X, Y, C, S, T, CT, SP	see page 3-55 or page 3-56
Discrete Outputs	F ffff	X, Y, C	

Whenever the Start input is energized, the drum's timer is enabled. As long as the event is true for the current step, the timer runs during that step. When the step count equals the counts per step, the drum transitions to the next step. This process stops when the last step is complete, or when the Reset input is energized. The drum enters the preset step chosen upon a CPU program-to-run mode transition, and whenever the Reset input is energized.

Drum instructions use four counters in the CPU. The ladder program can read the counter values for the drum's status. The ladder program may write a new preset step number to CTA(n+2) at any time. However, the other counters are for monitoring purposes only.

Counter Number	D2-250-1 Ranges of (n)	D2-260/D2-262 Ranges of (n)	Function	Counter Bit Function
CTA(n)	0–174	0–374	Counts in step	CT(n) = Drum Complete
CTA(n+1)	1–175	1–375	Timer value	CT(n+1) = (not used)
CTA(n+2)	2–176	2–376	Preset Step	CT(n+2) = (not used)
CTA(n+3)	3–177	3–377	Current Step	CT(n+3) = (not used)

The following ladder program shows the EDRUM instruction in a typical ladder program, as shown by *Direct*SOFT. Steps 1 through 11 are used, and all 16 output points are used. The preset step is step 1. The timebase runs at (K10 x 0.01) = 0.1 second per count. Therefore, the duration of step 1 is (1 x 0.1) = 0.1 second. Note that step 1 is time-based only (event is left blank). And, the output pattern for step 1 programs all outputs off, which is a typically desirable powerup condition. In the last rung, the Drum Complete bit (CT4) turns on output Y0 upon completion of the last step (step 11). A drum reset also resets CT4.



*Direct*SOFT Display

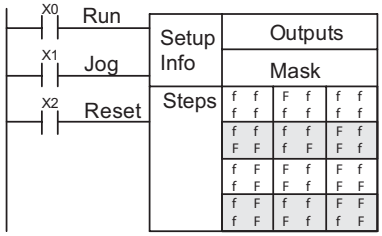
***NOTE**: If all events are true in an event only drum (a drum with 0 counts per step in all steps), the PLC completes one step of the drum per scan; thus, the drum will be complete in 16 scans. However, as the outputs of the drum are enabled any time the CPU is in RUN Mode, the drum discrete outputs will be energized as pulsed outputs for each scan.*

Handheld Programmer Drum Mnemonics

The EDRUM instruction can also be programmed using a Handheld Programmer. This section explains entry via the Handheld Programmer.

First, enter Store instructions for the ladder rungs controlling the drum's ladder inputs. In the example to the right, the timer drum's Start, Jog, and Reset inputs are controlled by X0, X1 and X2 respectively. The required keystrokes are listed beside the mnemonic.

These keystrokes precede the EDRUM instruction mnemonic. Note that the ladder rungs for Start, Jog, and Reset inputs are not limited to being single–contact rungs.



Handheld Programmer Keystrokes

Store X0 | $ STR | → | A 0 | ENT

(Repeat for Store X1 and Store X2)

Handheld Programmer Keystrokes

EDRUM CNT4 | SHFT | E 4 | D 3 | R ORN | U ISG | M ORST | → | E 4 | ENT

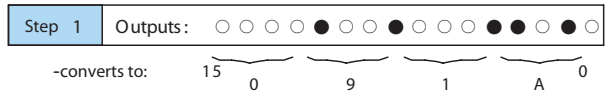
instructions, enter the EDRUM (using Counter CT0) as shown:

After entering the EDRUM mnemonic as above, the handheld programmer creates an input form for all the drum parameters. The input form consists of approximately fifty or more default mnemonic entries containing DEF (define) statements. The default mnemonics are already "input" for you, so they appear automatically. Use the NXT and PREV keys to move forward and backward through the form. Only the editing of default values is required, thus eliminating many keystrokes. The entries required for the basic timer drum are in the chart below.

***NOTE**: Default entries for output points and events are "DEF 0000", which means they are unassigned. If you need to go back and change an assigned output as unused again, enter "K0000". The entry will again show as "DEF 0000".*

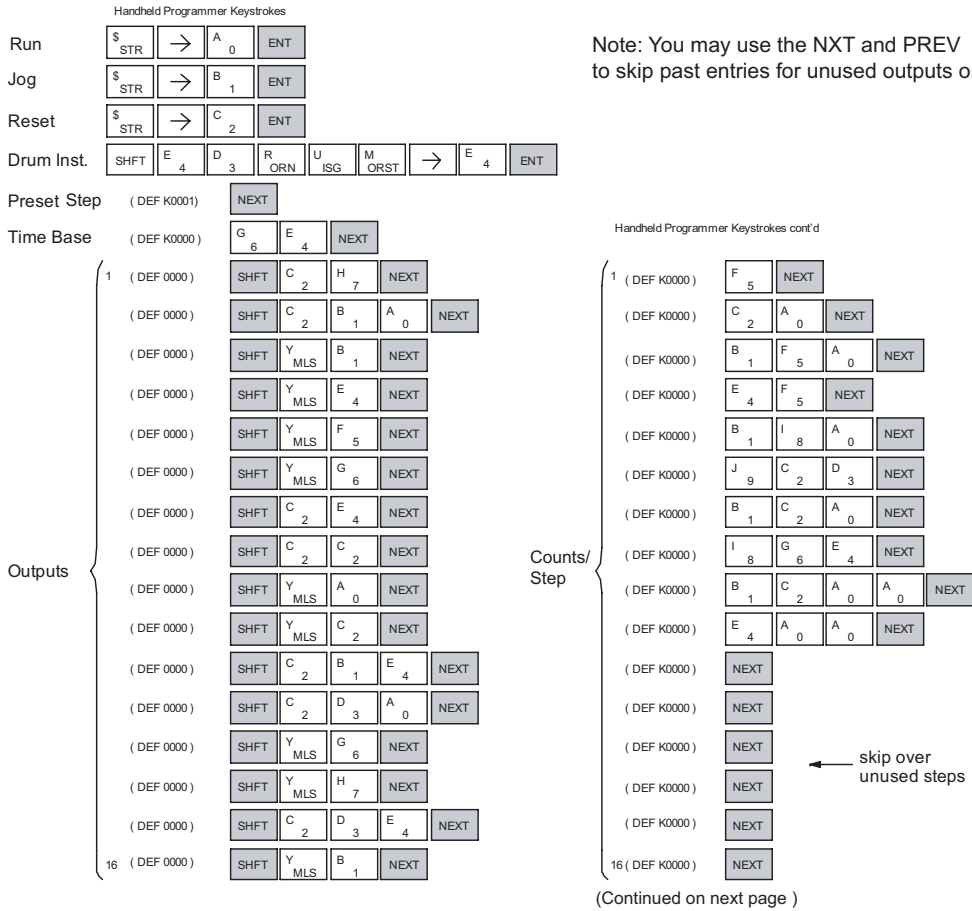
Drum Parameters	Multiple Entries	Mnemonic / Entry	Default Mnemonic	Valid Data Types	Ranges
Start Input	--	STR (plus input rung)	--	--	--
Jog Input	--	STR (plus input rung)	--	--	--
Reset Input	--	STR (plus input rung)	--	--	--
Drum Mnemonic	--	DRUM CNT aa	--	CT	0–174 (D2-250-1) 0–374 (D2-260/D2-262)
Step Preset	1	bb	DEF K0000	K	1–16
Timer base	1	cccc	DEF K0000	K	0–9999
Counts per step	16	dddd	DEF K0000	K	0–9999
Events	16	eeee	DEF K0000	X, Y, C, S, T, CT, SP	see either page 3-55 or page 3-56
Output points	16	ffff	DEF K0000	X, Y, C	
Output Mask	16	gggg	DEF K0000	K	0 - FFFF

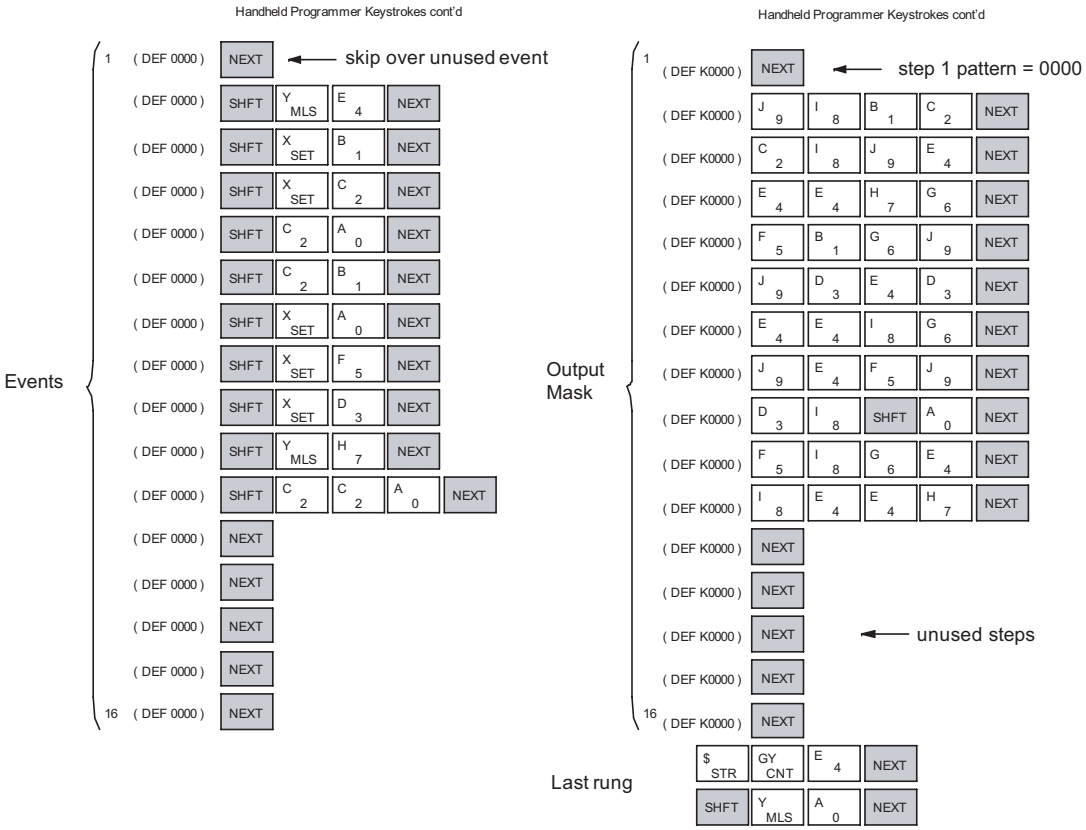
Using the DRUM entry chart (two pages before), we show the method of entry for the basic time/event drum instruction. First, we convert the output pattern for each step to the equivalent hex number, as shown in the following example.



The following diagram shows the method for entering the previous EDRUM example on the HPP. The default entries of the form are in parenthesis. After the drum instruction entry (on the fourth row), the remaining keystrokes over-write the numeric portion of each default DEF statement.

***NOTE**: Drum editing requires Handheld Programmer firmware version 2.21 or later.*





***NOTE**: Remember, you may use the NXT and PREV keys to skip past entries for unused outputs or steps.*

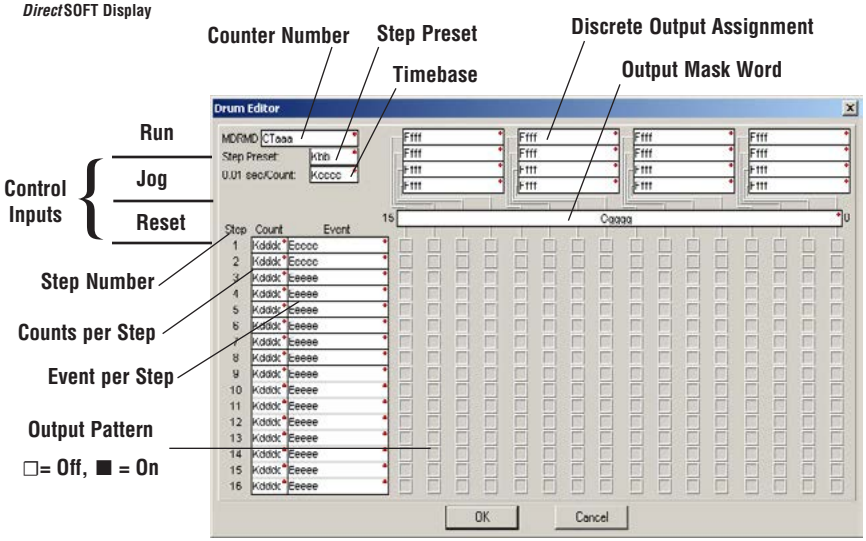
***NOTE:** For ease of operation when using the EDRUM instruction, we recommend using **Direct**SOFT over the handheld programmer.*

Masked Event Drum with Discrete Outputs (MDRMD)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The Masked Event Drum with Discrete Outputs has all the features of the basic Event Drum plus final output control for each step. It operates according to the general principles of drum operation covered in the beginning of this section. Below is the instruction in chart form as displayed by *Direct*SOFT.

*Direct*SOFT Display



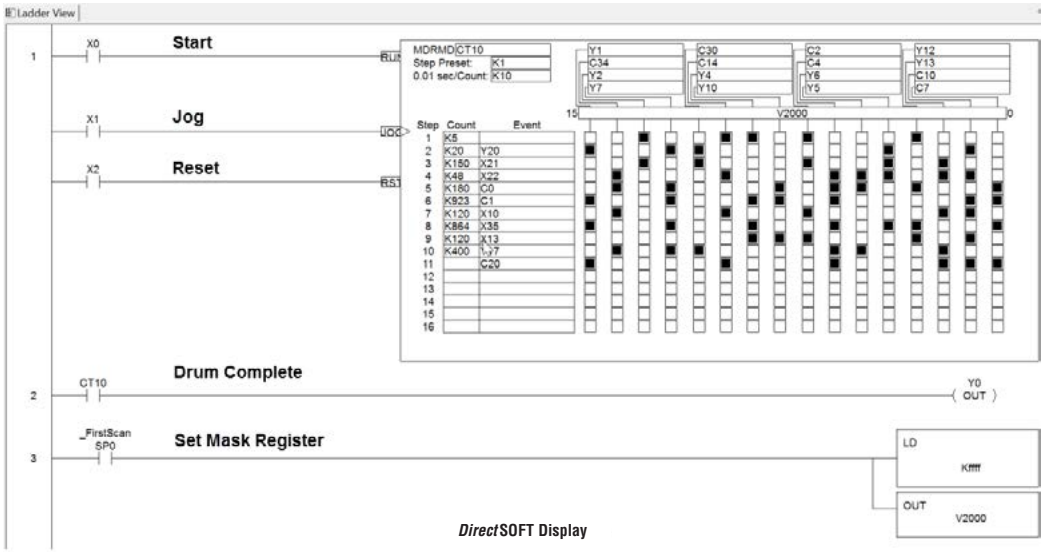
The Masked Event Drum with Discrete Outputs features 16 steps and 16 outputs. Drum outputs are logically ANDed bit-by-bit with an output mask word for each step. The Ggggg field specifies the beginning location of the 16 mask words. Step transitions occur on timed and/or event basis. The jog input also advances the step on each off-to-on transition. Time is specified in counts per step, and events are specified as discrete contacts. Unused steps and events can be left blank (this is the default entry). Whenever the Start input is energized, the drum's timer is enabled. As long as the event is true for the current step, the timer runs during that step. When the step count equals the counts per step, the drum transitions to the next step. This process stops when the last step is complete, or when the Reset input is energized. The drum enters the preset step chosen upon a CPU program-to-run mode transition, and whenever the Reset input is energized.

Drum Parameters	Field	Data Types	Ranges
Counter Number	aaa	–	0–174 (D2-250-1) 0–374 (D2-260/D2-262)
Step Preset	bb	K	1–16
Timer base	cccc	K	0–99.99 seconds
Counts per step	dddd	K	0–9999
Event	eeee	X, Y, C, S, T, ST, GX, GY, CT, SP	see either page 3-55 or page 3-56
Discrete Outputs	Ffff	X, Y, C, GX, GY	
Output Mask	Ggggg	V	

Drum instructions use four counters in the CPU. The ladder program can read the counter values for the drum's status. The ladder program may write a new preset step number to CTA(n+2) at any time. However, the other counters are for monitoring purposes only.

Counter Number	D2-250-1 Ranges of (n)	D2-260/D2-262 Ranges of (n)	Function	Counter Bit Function
CTA(n)	0–174	0–374	Counts in step	CT(n) = Drum Complete
CTA(n+1)	1–175	1–375	Timer value	CT(n+1) = (not used)
CTA(n+2)	2–176	2–376	Preset Step	CT(n+2) = (not used)
CTA(n+3)	3–177	3–377	Current Step	CT(n+3) = (not used)

The following ladder program shows the MDRMD instruction in a typical ladder program, as shown by ***Direct***SOFT. Steps 1 through 11 are used, and all 16 output points are used. The output mask word is at V2000. The final drum outputs are shown above the mask word as individual bits. The data bits in V2000 are logically ANDed with the output pattern of the current step in the drum. If you want all drum outputs to be off after powerup, write zeros to V2000 on the first scan. Ladder logic may update the output mask at any time to enable or disable the drum outputs The preset step is step 1. The timebase runs at (K10 x 0.01) = 0.1 second per count. Therefore, the duration of step 1 is (5 x 0.1) = 0.5 seconds. Note that step 1 is time-based only (event is left blank). In the last rung, the Drum Complete bit (CT10) turns on output Y0 upon completion of the last step (step 11). A drum reset also resets CT10.



***Direct*SOFT Display**

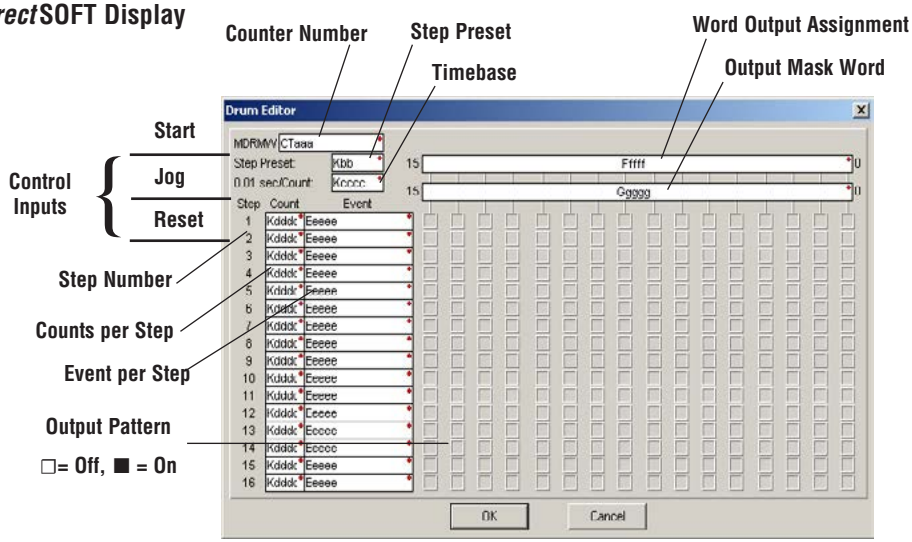
***NOTE**: The ladder program must load constants in V2000 through V2012 to cover all mask registers for the eleven steps used in this drum.*

Masked Event Drum with Word Output (MDRMW)

☒ 230
☒ 240
☑ 250-1
☑ 260
☑ 262

The Masked Event Drum with Word Output features outputs organized as bits of a single word, rather than discrete points. It operates according to the general principles of drum operation covered in the beginning of this section. Below is the instruction in chart form as displayed by *Direct*SOFT.

*Direct*SOFT Display



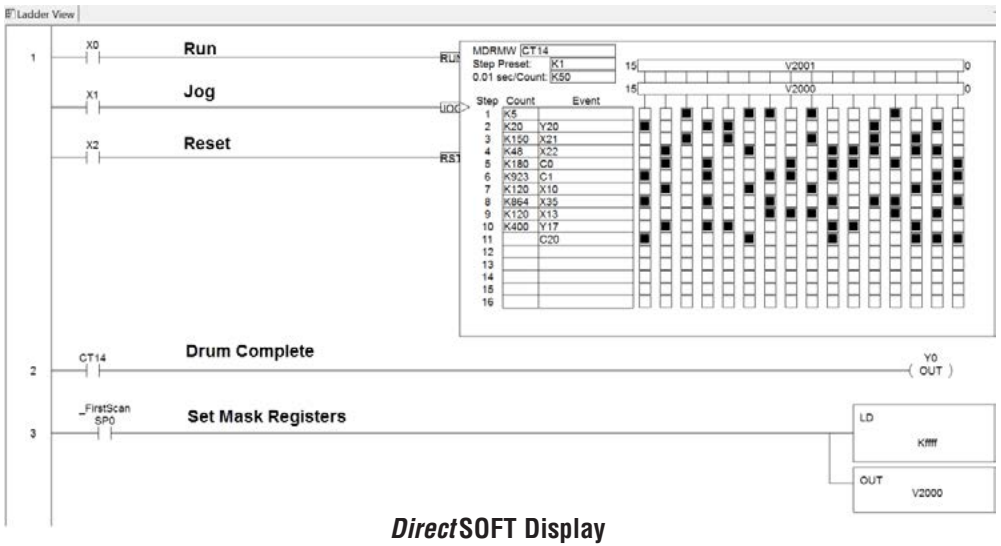
The Masked Event Drum with Word Output features 16 steps and 16 outputs. Drum outputs are logically ANDed bit-by-bit with an output mask word for each step. The Ggggg field specifies the beginning location of the 16 mask words, creating the final output (Ffff field). Step transitions occur on a timed and/or event basis. The jog input also advances the step on each off-to-on transition. Time is specified in counts per step, and events are specified as discrete contacts. Unused steps and events can be left blank (this is the default entry). Whenever the Start input is energized, the drum's timer is enabled. As long as the event is true for the current step, the timer runs during that step. When the step count equals the counts per step, the drum transitions to the next step. This process stops when the last step is complete, or when the Reset input is energized. The drum enters the preset step chosen upon a CPU program-to-run mode transition, and whenever the Reset input is energized.

Drum Parameters	Field	Data Types	Ranges
Counter Number	aaa	–	0–174 (D2-250-1) 0–374 (D2-260/D2-262)
Preset Step	bb	K	1–16
Timer base	cccc	K	0–99.99 seconds
Counts per step	dddd	K	0–9999
Event	eeee	X, Y, C, S, T, CT, GX, GY, SP	see either page 3-55 or page 3-56
Word Output	Fffff	V	
Output Mask	Ggggg	V	

Drum instructions use four counters in the CPU. The ladder program can read the counter values for the drum's status. The ladder program may write a new preset step number to CTA(n+2) at any time. However, the other counters are for monitoring purposes only.

Counter Number	D2-250-1 Ranges of (n)	D2-260/D2-262 Ranges of (n)	Function	Counter Bit Function
CTA(n)	0–174	0–374	Counts in step	CT(n) = Drum Complete
CTA(n+1)	1–175	1–375	Timer value	CT(n+1) = (not used)
CTA(n+2)	2–176	2–376	Preset Step	CT(n+2) = (not used)
CTA(n+3)	3–177	3–377	Current Step	CT(n+3) = (not used)

The following ladder program shows the MDRMW instruction in a typical ladder program, as shown by *Direct*SOFT. Steps 1 through 11 are used, and all sixteen output points are used. The output mask word is at V2000. The final drum outputs are shown above the mask word as a word at V2001. The data bits in V2000 are logically ANDed with the output pattern of the current step in the drum, generating the contents of V2001. If you want all drum outputs to be off after powerup, write zeros to V2000 on the first scan. Ladder logic may update the output mask at any time to enable or disable the drum outputs. The preset step is step 1. The timebase runs at (K50 x 0.01) = 0.5 seconds per count. Therefore, the duration of step 1 is (5 x 0.5) = 2.5 seconds. Note that step 1 is time-based only (event is left blank). In the last rung, the Drum Complete bit (CT14) turns on output Y0 upon completion of the last step (step 11). A drum reset also resets CT14.



***Direct*SOFT Display**

***NOTE**: The ladder program must load constants in V2000 through V2012 to cover all mask registers for the 11 steps used in this drum.*

RLL^{PLUS} Stage Programming

In This Chapter...

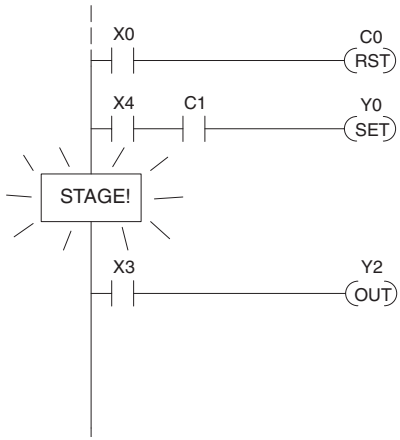
Introduction to Stage Programming

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

Stage Programming (available in all DL205 CPUs) provides a way to organize and program complex applications with relative ease, when compared to purely relay ladder logic (RLL) solutions. Stage programming does not replace or negate the use of traditional boolean ladder programming. This is why Stage Programming is also called RLL^{PLUS}. You will not have to discard any training or experience you already have. Stage programming simply allows you to divide and organize a RLL program into groups of ladder instructions called stages. This allows quicker and more intuitive ladder program development than traditional RLL alone provides.

Overcoming "Stage Fright"

- Many PLC programmers in the industry have become comfortable using RLL for every PLC program they write... but often remain skeptical or even fearful of learning new techniques such as stage programming. While RLL is great at solving boolean logic relationships, it has disadvantages as well:
- Large programs can become almost unmanageable, because of a lack of structure.
- In RLL, latches must be tediously created from self-latching relays.
- When a process gets stuck, it is difficult to find the rung where the error occurred.
- Programs become difficult to modify later, because they do not intuitively resemble the application problem they are solving.



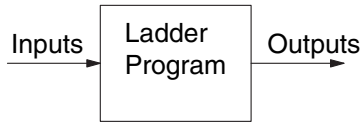
It's easy to see that these inefficiencies consume a lot of additional time, and time is money. *Stage programming overcomes these obstacles!* We believe a few moments of studying the stage concept is one of the greatest investments in programming speed and efficiency a PLC programmer can make!

So, we encourage you to study stage programming and add it to your "toolbox" of programming techniques. This chapter is designed as a self-paced tutorial on stage programming. For best results:

- Start at the beginning and do not skip over any sections.
- Study each stage programing concept by working through each example. The examples build progressively on each other.
- Read the Stage Questions and Answers at the end of the chapter for a quick review.

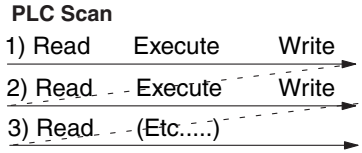
Learning to Draw State Transition Diagrams

Introduction to Process States



Those familiar with ladder program execution know the CPU must scan the ladder program repeatedly, over and over. Its three basic steps are:

1. Read the inputs.
2. Execute the ladder program.
3. Write the outputs.



The benefit is that a change at the inputs can affect the outputs in just a few milliseconds.

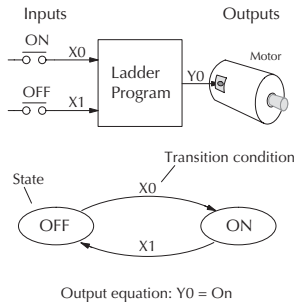
Most manufacturing processes consist of a series of activities or conditions, each lasting for several seconds, minutes, or even hours. We might call these "process states," which are either active or inactive at any particular time. A challenge for RLL programs is that a particular input event may last for a brief instant. We typically create latching relays in RLL to preserve the input event in order to maintain a process state for the required duration. We can organize and divide ladder logic into sections called "stages," representing process states. But before we describe stages in detail, we will reveal **the secret to understanding stage programming:** state transition diagrams.

The Need for State Diagrams

Sometimes we need to forget about the scan nature of PLCs, and focus our thinking toward the states of the process we need to identify. Clear thinking and concise analysis of an application gives us the best chance at writing efficient, bug-free programs. *State diagrams are tools to help us draw a picture of our process!* You will discover that if we can get the picture right, **our program will also be right!**

A 2–State Process

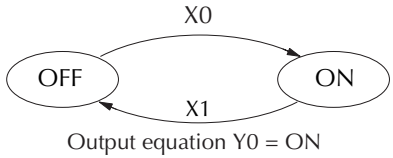
Consider the simple process shown to the right, which controls an industrial motor. We will use a green momentary SPST pushbutton to turn the motor on, and a red one to turn it off. The machine operator will press the appropriate pushbutton for a second or so. The two states of our process are ON and OFF.



The next step is to draw a *state transition diagram*, as shown to the right. It shows the two states OFF and ON, with two transition lines in-between. When the event X0 is true, we transition from OFF to ON. When X1 is true, we transition from ON to OFF.

If you're following along, you are very close to grasping the concept and the problem-solving power of state transition diagrams. The output of our controller is Y0, which is true any time we are in the ON state. In a boolean sense, Y0=ON state.

The state transition diagram to the right is a picture of the solution we need to create. The beauty of it is this: it expresses the problem independently of the programming language we may use to realize it. In other words, *by drawing the diagram we have already solved the control problem!*

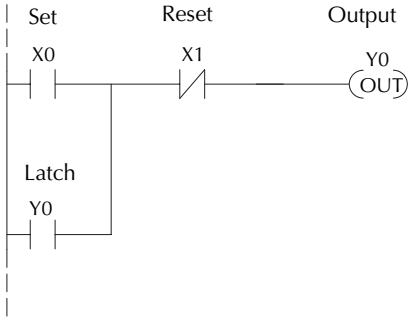


First, we'll translate the state diagram to traditional RLL. Then we'll show how easy it is to translate the diagram into a stage programming solution.

RLL Equivalent

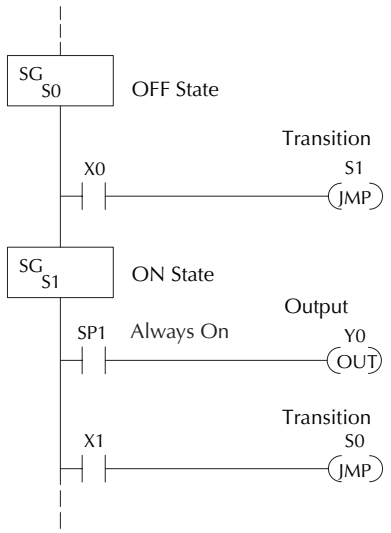
The RLL solution is shown to the right. It consists of a self-latching motor output coil, Y0. When the On pushbutton (X0) is pressed, output coil Y0 turns on and the Y0 contact on the second row latches itself on. So, X0 **sets the latch** Y0 on, and it remains on after the X0 contact opens.

When the Off pushbutton (X1) is pressed, it opens the normally-closed X1 contact, which **resets the latch**. Motor output Y0 turns off.



Stage Equivalent

The stage program solution is shown to the right. The two inline stage boxes S0 and S1 correspond to the two states OFF and ON. The ladder rung(s) below each stage box belong to each respective stage. This means that *the PLC only has to scan those rungs when the corresponding stage is active!*



For now, let's assume we begin in the OFF State, so stage S0 is active. When the On pushbutton (X0) is pressed, a stage transition occurs. The JMP S1 instruction executes, which simply turns off the Stage bit S0 and turns on Stage bit S1. So on the next PLC scan, the CPU will not execute Stage S0, but will execute stage S1!

In the On State (Stage S1), we want the motor to always be on. The special relay contact SP1 is defined as always on, so Y0 turns the motor on.

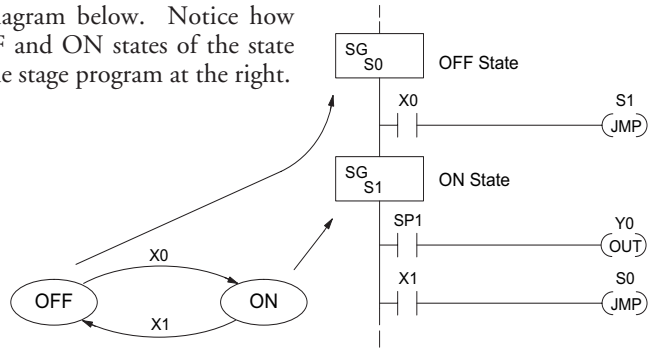
When the Off pushbutton (X1) is pressed, a transition back to the Off State occurs. The JMP S0 instruction executes, which simply turns off the Stage bit S1 and turns on Stage bit S0. On the next PLC scan, the CPU will not execute Stage S1, so the motor output Y0 will turn off. The Off state (Stage 0) will be ready for the next cycle.

Let's Compare

Right now, you may be thinking "I don't see the big advantage to Stage Programming... in fact, the stage program is longer than the plain RLL program". Well, now is the time to exercise a bit of faith. As control problems grow in complexity, stage programming quickly out-performs RLL in simplicity, program size, etc.

For example, consider the diagram below. Notice how easy it is to correlate the OFF and ON states of the state transition diagram below to the stage program at the right.

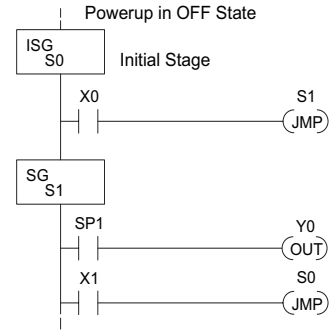
Now, we challenge anyone to easily identify the same states in the RLL program on the previous page!



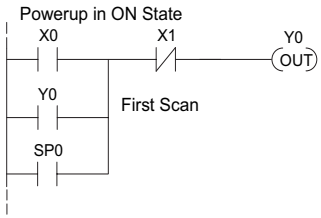
Initial Stages

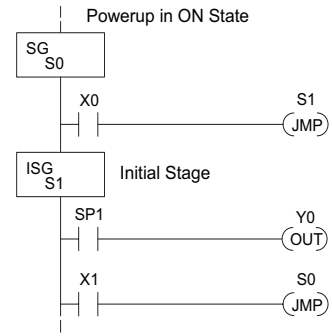
At powerup and Program-to-Run Mode transitions, the PLC always begins with all normal stages (SG) off. So, the stage programs shown so far have actually had no way to get started (because rungs are not scanned unless their stage is active).

Assume that we want to always begin in the Off state (motor off), which is how the RLL program works. The Initial Stage (ISG) is defined to be active at powerup. In the modified program to the right, we have changed stage S0 to the ISG type. This ensures the PLC will scan contact X0 after powerup, because Stage S0 is active. **After powerup, an Initial Stage (ISG) works just like any other stage!**



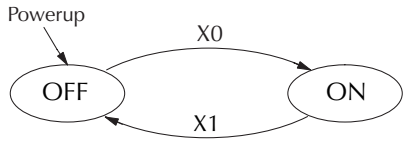
We can change both programs so that the motor is ON at powerup. In the RLL below, we must add a first scan relay SP0, latching Y0 on. In the stage example to the right, we simply make Stage S1 an initial stage (ISG) instead of S0.





NOTE: *If the ISG is within the retentive range for stages, the ISG will remain in the state it was in before power down and will NOT turn itself on during the first scan.*

We can mark our desired powerup state as shown to the right, which helps us remember to use the appropriate Initial Stages when creating a stage program. It is permissible to have as many initial stages as the process requires.

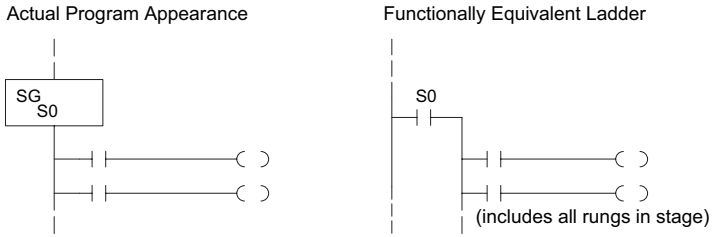


What Stage Bits Do

You may recall that a stage is a section of ladder program which is either active or inactive at a given moment. All stage bits (S0 to Sxxx) reside in the PLC's image register as individual status bits. Each stage bit is either a boolean 0 or 1 at any time.

Program execution always reads ladder rungs from top to bottom, and from left to right. The drawing below shows the effect of stage bit status. The ladder rungs below the stage instruction continuing until the next stage instruction or the end of program belong to stage 0. Its equivalent operation is shown on the right. When S0 is true, the two rungs have power flow.

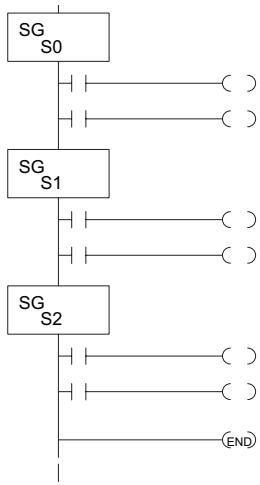
- If Stage bit S0 = 0, its ladder rungs *are not scanned* (executed).
- If Stage bit S0 = 1, its ladder rungs *are scanned* (executed).



Stage Instruction Characteristics

The inline stage boxes on the left power rail divide the ladder program rungs into stages. Some stage rules are:

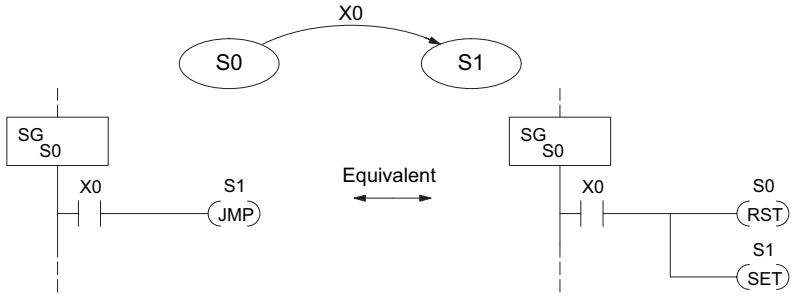
- **Execution** – Only logic in active stages are executed on any scan.
- **Transitions** – Stage transition instructions take effect on the next occurrence of the stages involved.
- **Octal numbering** – Stages are numbered in octal, like I/O points, etc. So "S8" is not valid.
- **Total Stages** – The maximum number of stages is CPU dependent.
- **No duplicates** – Each stage number is unique and can be used just once.
- **Any order** – You can skip numbers and sequence the stage numbers in any order.
- **Last Stage** – The last stage in the ladder program includes all rungs from its stage box until the end coil.



Using the Stage Jump Instruction for State Transitions

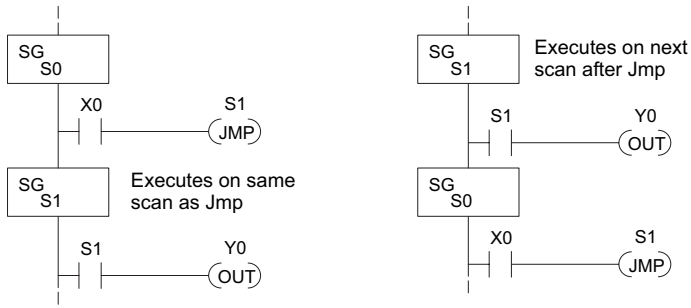
Stage Jump, Set, and Reset Instructions

The Stage JMP instruction we have used deactivates the stage in which the instruction occurs, while activating the stage in the JMP instruction. Refer to the state transition shown below. When contact X0 energizes, the state transition from S0 to S1 occurs. The two stage examples shown below are equivalent. So, the Stage Jump instruction is equal to a Stage Reset of the current stage, plus a Stage Set instruction for the stage to which we want to transition.



Please Read Carefully – The jump instruction is easily misunderstood. The "jump" does not occur immediately like a GOTO or GOSUB program control instruction when executed. Here's how it works:

- The jump instruction resets the stage bit of the stage in which it occurs. All rungs in the stage still finish executing during the current scan, *even if there are other rungs in the stage below the jump instruction!*
- The reset will be in effect on the following scan, so the stage that executed the jump instruction previously will be inactive and bypassed.
- The stage bit of the stage named in the Jump instruction will be set immediately, so the stage will be executed on its next occurrence. In the left program shown below, stage S1 executes during the same scan as the JMP S1 occurs in S0. In the example on the right, Stage S1 executes on the next scan after the JMP S1 executes, because stage S1 is located above stage S0.

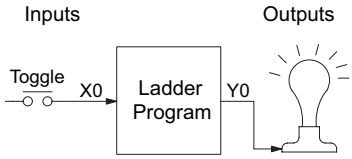


***NOTE**: Assume we start with Stage 0 active and Stage 1 inactive for both examples.*

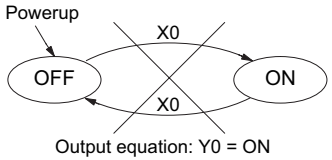
Stage Program Example: Toggle On/Off Lamp Controller

A 4-State Process

In the process shown to the right, we use an ordinary momentary pushbutton to control a light bulb. The ladder program will latch the switch input, so that we will push and release to turn on the light, push and release again to turn it off (sometimes called toggle function). Sure, we could buy a mechanical switch with the alternate on/off action built in... However, this example is educational and also fun!

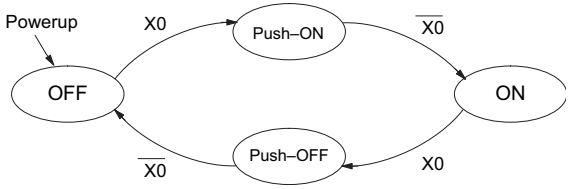


Next we draw the state transition diagram. A typical first approach is to use X0 for both transitions (like the example shown to the right). However, *this is incorrect* (please keep reading).



Note that this example differs from the motor example, because now we have only one pushbutton. When we press the pushbutton, both transition conditions are met. We would transition around the state diagram at top speed. If implemented in Stage, this solution would flash the light on or off each scan (obviously undesirable)!

The solution is to make the push and the release of the pushbutton separate events. Refer to the new state transition diagram below. At powerup we enter the OFF state. When switch X0 is pressed, we enter the Press-ON state. When it is released, we enter the ON state. Note that X0 with the bar above it denotes X0 NOT.

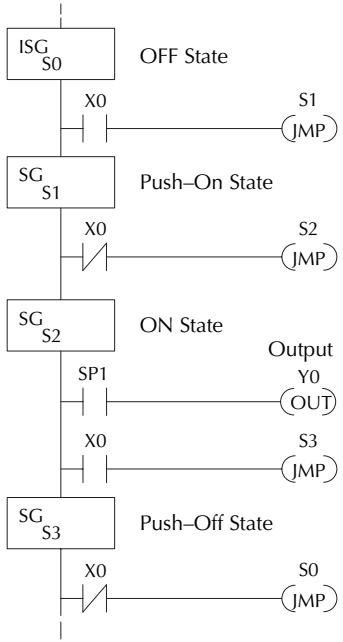


Output equation: Y0 = ON

When in the ON state, another push and release cycle similarly takes us back to the OFF state. Now we have two unique states (OFF and ON) used when the pushbutton is released, which is what was required to solve the control problem.

The equivalent stage program is shown to the right. The desired powerup state is OFF, so we make S0 an initial stage (ISG). In the ON state, we add special relay contact SP1, which is always on.

Note that even as our programs grow more complex, it is still easy to correlate the state transition diagram with the stage program!



Four Steps to Writing a Stage Program

By now, you've probably noticed that we follow the same steps to solve each example problem. The steps will probably come to you automatically if you work through all the examples in this chapter. It's helpful to have a checklist to guide us through the problem solving. The following steps summarize the stage program design procedure:

1. Write a Word Description of the application.

Describe all functions of the process in your own words. Start by listing what happens first, then next, etc. If you find there are too many things happening at once, try dividing the problem into more than one process. Remember, you can still have the processes communicate with each other to coordinate their overall activity.

2. Draw the Block Diagram.

Inputs represent all the information the process needs for decisions, and outputs connect to all devices controlled by the process.

- Make lists of inputs and outputs for the process.
- Assign I/O point numbers (X and Y) to physical inputs and outputs.

3. Draw the State Transition Diagram.

The state transition diagram describes the central function of the block diagram, reading inputs and generating outputs.

- Identify and name the states of the process.
- Identify the event(s) required for each transition between states.
- Ensure the process has a way to re-start itself, or is cyclical.
- Choose the powerup state for your process.
- Write the output equations.

4. Write the Stage Program.

Translate the state transition diagram into a stage program.

- Make each state a stage. Remember to number stages in octal. Up to 256 total stages are available in the D2-230 CPU. Up to 512 total stages are available in the D2-240 CPU. Up to 1024 total stages are available in the D2-250–1, D2-260 and D2-262 CPUs.
- Put transition logic inside the stage which originates each transition (the stage each arrow points away from).
- Use an initial stage (ISG) for any states that must be active at powerup.
- Place the outputs or actions in the appropriate stages.

You will notice that Steps 1 through 3 prepare us to write the stage program in Step 4. However, the program virtually writes itself because of the preparation beforehand. Soon you will be able to start with a word description of an application and create a stage program in one easy session!

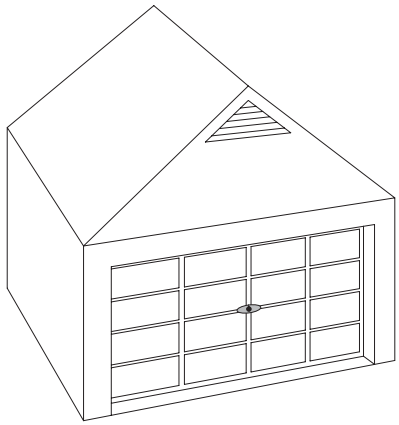
Stage Program Example: A Garage Door Opener

Garage Door Opener Example

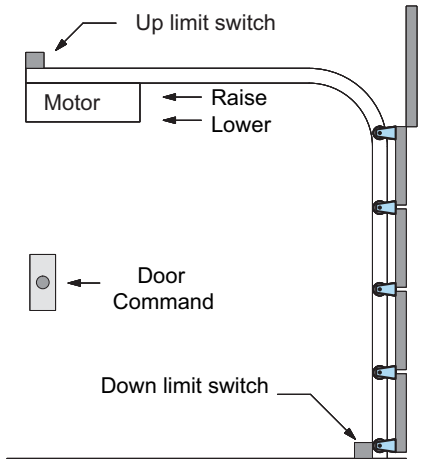
In this next stage programming example we will create a garage door opener controller. Hopefully most readers are familiar with this application, and we can have fun besides!

The first step we must take is to describe how the door opener works. We will start by achieving the basic operation, waiting to add extra features later (stage programs are very easy to modify).

Our garage door controller has a motor which raises or lowers the door on command. The garage owner pushes and releases a momentary pushbutton once to raise the door. After the door is up, another push-release cycle will lower the door.



In order to identify the inputs and outputs of the system, it's sometimes helpful to sketch its main components, as shown in the door side view to the right. The door has an up limit and a down limit switch. Each limit switch closes only when the door has reached the end of travel in the corresponding direction. In the middle of travel, neither limit switch is closed.

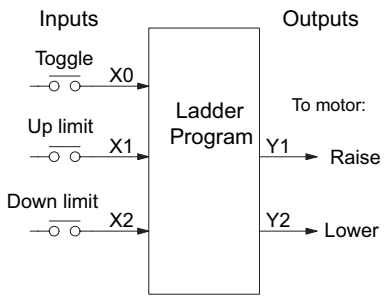


The motor has two command inputs: raise and lower. When neither input is active, the motor is stopped.

The door command is a simple pushbutton. Whether wall-mounted as shown, or a radio-remote control, all door control commands logically OR together as one pair of switch contacts.

Draw the Block Diagram

The block diagram of the controller is shown to the right. Input X0 is from the pushbutton door control. Input X1 energizes when the door reaches the full up position. Input X2 energizes when the door reaches the full down position. When the door is positioned between fully up or down, both limit switches are open.

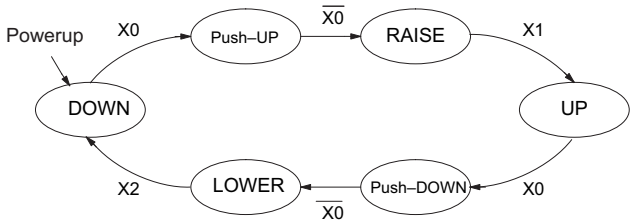


The controller has two outputs to drive the motor. Y1 is the up (raise the door) command, and Y2 is the down (lower the door) command.

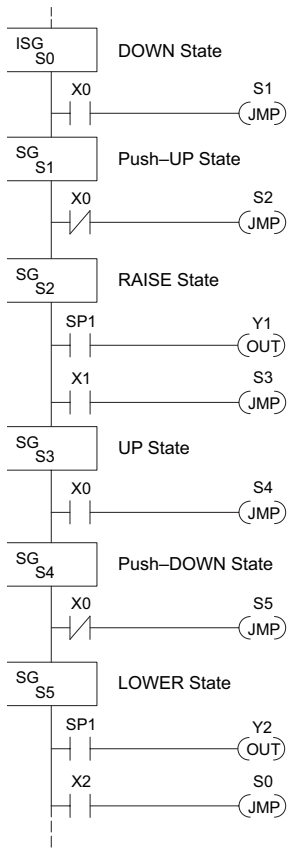
Draw the State Diagram

Now we are ready to draw the state transition diagram. Like the previous light bulb controller example, this application also has only one switch for the command input. Refer to the figure below.

- When the door is down (DOWN state), nothing happens until X0 energizes. Its push and release brings us to the RAISE state, where output Y1 turns on and causes the motor to raise the door.
- We transition to the UP state when the up limit switch (X1) energizes, and turns off the motor.
- Then nothing happens until another X0 press-release cycle occurs. That takes us to the LOWER state, turning on output Y2 to command the motor to lower the door. We transition back to the DOWN state when the down limit switch (X2) energizes.



Output equations: Y1 = Raise Y2 = Lower



The equivalent stage program is shown to the right. For now, we will assume the door is down at powerup, so the desired powerup state is DOWN. We make S0 an initial stage (ISG). Stage S0 remains active until the door control pushbutton activates. Then we transition (JMP) to Push-UP stage, S1.

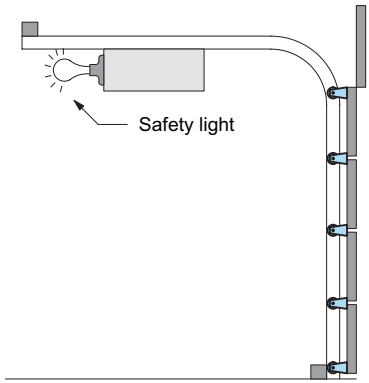
A push-release cycle of the pushbutton takes us through stage S1 to the RAISE stage, S2. We use the always-on contact SP1 to energize the motor's raise command, Y1. When the door reaches the fully-raised position, the up limit switch X1 activates. This takes us to the UP Stage S3, where we wait until another door control command occurs.

In the UP Stage S3, a push-release cycle of the pushbutton will take us to the LOWER Stage S5, where we activate Y2 to command the motor to lower the door. This continues until the door reaches the down limit switch, X2. When X2 closes, we transition from Stage S5 to the DOWN stage S0, where we began.

NOTE: *The only thing special about an initial stage (ISG) is that it is automatically active at powerup. Afterwards, it is just like any other.*

Add Safety Light Feature

Next we will add a safety light feature to the door opener system. It's best to get the main function working first as we have done, then adding the secondary features.

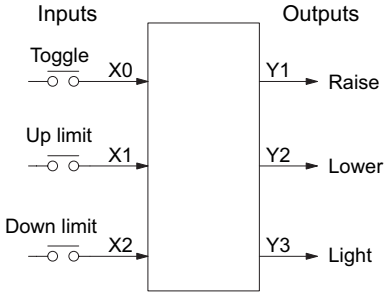


The safety light is standard on many commercially-available garage door openers. It is shown to the right, mounted on the motor housing. The light turns on upon any door activity, remaining on for approximately 3 minutes afterwards.

This part of the exercise will demonstrate the use of parallel states in our state diagram. Instead of using the JMP instruction, we will use the set and reset commands.

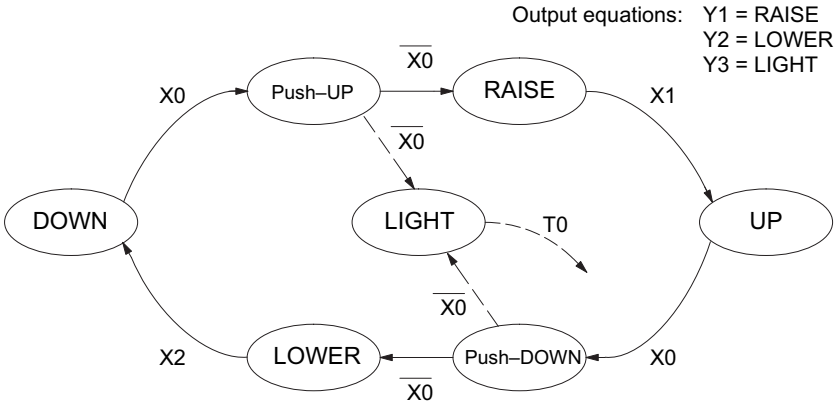
Modify the Block Diagram and State Diagram

To control the light bulb, we add an output to our controller block diagram, shown to the right, Y3 is the light control output.



In the diagram below, we add a state called "LIGHT". Whenever the garage owner presses the door control switch and releases, the RAISE or LOWER state is active *and the LIGHT state is simultaneously active*. The line to the Light state is dashed, because it is not the primary path.

We can think of the Light state as a parallel process to the raise and lower state. The paths to the Light state are not a transition (Stage JMP), but a State Set command. In the logic of the Light stage, we will place a three-minute timer. When it expires, timer bit T0 turns on and resets the Light stage. The path out of the Light stage goes nowhere, indicating the Light stage becomes inactive, and the light goes out!



Using a Timer Inside a Stage

The finished modified program is shown to the right. The shaded areas indicate the program additions.

In the Push-UP stage S1, we add the Set Stage Bit S6 instruction. When contact X0 opens, we transition from S1 and go to two new active states: S2 and S6. In the Push-DOWN state S4, we make the same additions. So, any time someone presses the door control pushbutton, the light turns on.

Most new stage programmers would be concerned about where to place the Light Stage in the ladder, and how to number it. The good news is that it doesn't matter!

- Choose an unused Stage number, and use it for the new stage and as the reference from other stages.
- Placement in the program is not critical, so we place it at the end.

You might think that each stage has to be directly under the stage that transitions to it. While it is good practice, it is not required (that's good, because our two locations for the Set S6 instruction make that impossible). Stage numbers and how they are used determines the transition paths.

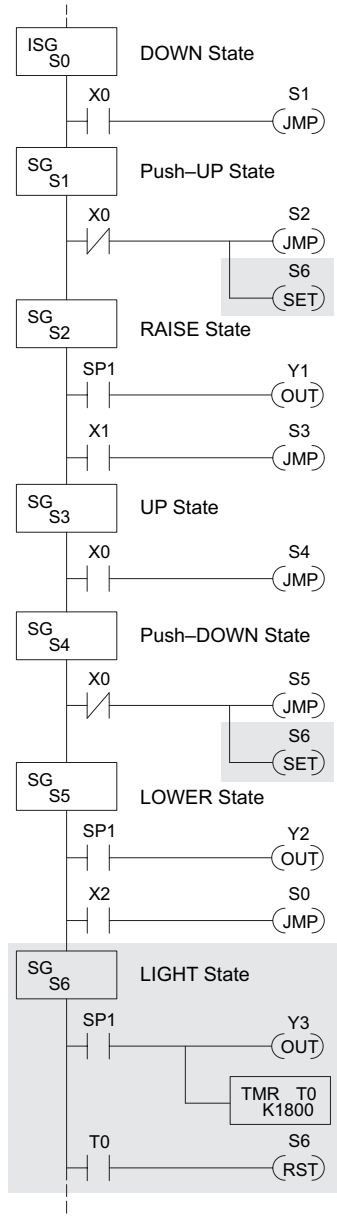
In stage S6, we turn on the safety light by energizing Y3. Special relay contact SP1 is always on. Timer T0 times at 0.1 second per count. To achieve 3 minutes time period, we calculate:

$$K = \frac{3 \text{ min. x } 60 \text{ sec/min}}{0.1 \text{ sec/count}}$$

$$K = 1800 \text{ counts}$$

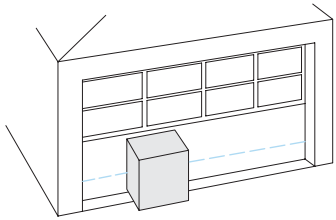
The timer has power flow whenever stage S6 is active. The corresponding timer bit T0 is set when the timer expires. So three minutes later, T0=1 and the instruction Reset S6 causes the stage to be inactive.

While Stage S6 is active and the light is on, stage transitions in the primary path continue normally and independently of Stage 6. That is, the door can go up, down, or whatever, but the light will be on for precisely 3 minutes.

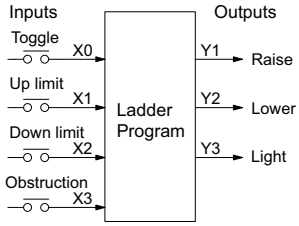


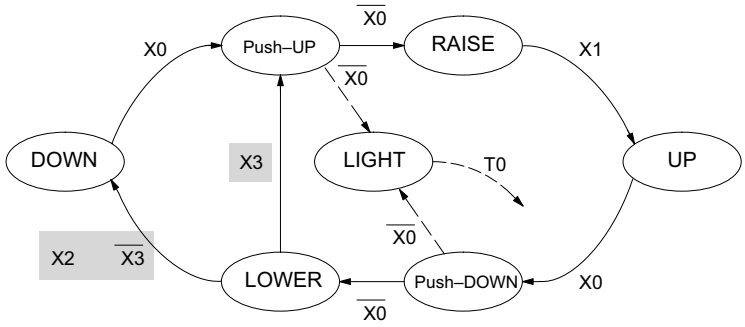
Add Emergency Stop Feature

Some garage door openers today will detect an object under the door. This halts further lowering of the door. Usually implemented with a photocell ("electric-eye"), a door in the process of being lowered will halt and begin raising. We will define our safety feature to work in this way, adding the input from the photocell to the block diagram as shown to the right. X3 will be on if an object is in the path of the door.



Next, we make a simple addition to the state transition diagram, shown in shaded areas in the figure below. Note the new transition path at the top of the LOWER state. If we are lowering the door and detect an obstruction (X3), we then jump to the Push-UP State. We do this instead of jumping directly to the RAISE state, to give the Lower output Y2 one scan to turn off, before the Raise output Y1 energizes.

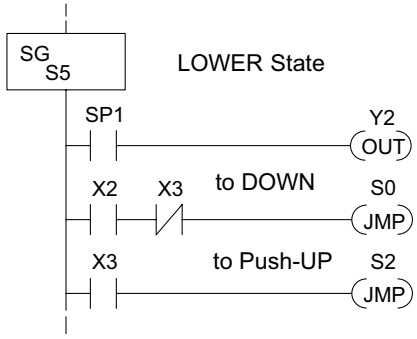




Exclusive Transitions

It is theoretically possible the down limit (X2) and the obstruction input (X3) could energize at the same moment. In that case, we would "jump" to the Push-UP and DOWN states simultaneously, which does not make sense.

Instead, we give priority to the obstruction by changing the transition condition to the DOWN state to [X2 AND NOT X3]. This ensures the obstruction event has the priority. The modifications we must make to the LOWER Stage (S5) logic are shown to the right. The first rung remains unchanged. The second and third rungs implement the transitions we need. Note the opposite relay contact usage for X3, which ensures the stage will execute only one of the JMP instructions.

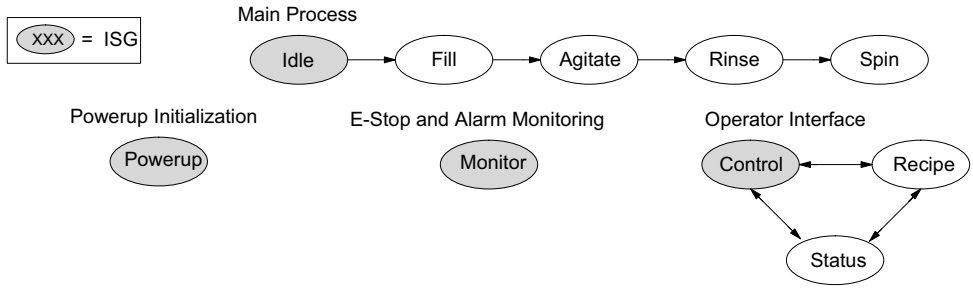


Stage Program Design Considerations

Stage Program Organization

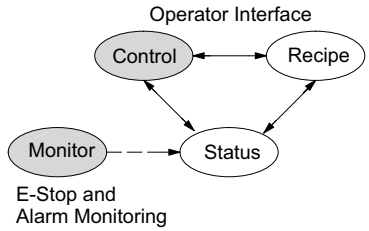
The examples so far in this chapter used one self-contained state diagram to represent the main process. However, we can have multiple processes implemented in stages, all in the same ladder program. New stage programmers sometimes try to turn a stage on and off each scan, based on the false assumption that only one stage can be on at a time. For ladder rungs that you want to execute each scan, put them in a stage that is always on.

The following figure shows a typical application. During operation, the primary manufacturing activity Main Process, Powerup Initialization, E-Stop and Alarm Monitoring, and Operator Interface are all running. At powerup, four initial stages shown begin operation.



In a typical application, the separate stage sequences above operate as follows:

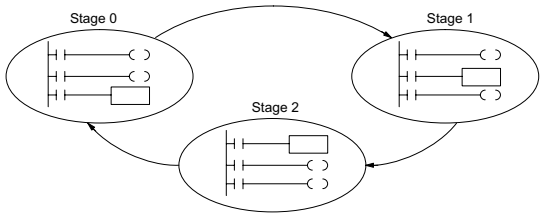
- **Powerup Initialization** – This stage contains ladder rung tasks performed once at powerup. Its last rung resets the stage, so this stage is only active for one scan (or only as many scans that are required)
- **Main Process** – This stage sequence controls the heart of the process or machine. One pass through the sequence represents one part cycle of the machine, or one batch in the process.
- **E-Stop and Alarm Monitoring** – This stage is always active because it is watching for errors that could indicate an alarm condition or require an emergency stop. It is common for this stage to reset stages in the main process or elsewhere, in order to initialize them after an error condition.
- **Operator Interface** – This is another task that must always be active and ready to respond to an operator. It allows an operator interface to change modes, etc., independent of the current main process step.



Although we have separate processes, there can be coordination among them. For example, in an error condition, the Status Stage may want to automatically switch the operator interface to the status mode to show error information as shown to the right. The monitor stage could set the stage bit for Status and Reset the stages Control and Recipe.

How Instructions Work Inside Stages

We can think of states or stages as simply dividing our ladder program as depicted in the figure below. Each stage contains only the ladder rungs which are needed for the corresponding state of the process. The logic for transitioning out of a stage is contained within that stage. It's easy to choose which ladder rungs are active at powerup by using an "initial" stage type (ISG).



Most instructions work like they do in standard RLL. You can think of a stage like a miniature RLL program that is either active or inactive.

Output Coils – As expected, output coils in active stages will turn on or off outputs according to power flow into the coil. However, note the following:

- Outputs work as usual, provided each output reference (such as "Y3") is used in only one stage.
- Output coils automatically turn off when leaving a stage. However, Set and Reset instructions are not "undone" when leaving a stage.
- An output can be referenced from more than one stage, as long as only one of the stages is active at a time.
- If an output coil is controlled by more than one stage simultaneously, the active stage nearest the bottom of the program determines the final output status during each scan. So, use the OROUT instruction instead when you want multiple stages to have a logical OR control of an output.

One-Shot or PD coils – Use care if you must use a Positive Differential coil in a stage. Remember the input to the coil must make a 0–1 transition. If the coil is already energized on the first scan when the stage becomes active, the PD coil will not work. This is because the 0–1 transition did not occur.

PD coil alternative: If there is a task that you want to do only once (on 1 scan), it can be placed in a stage that transitions to the next stage on the same scan.

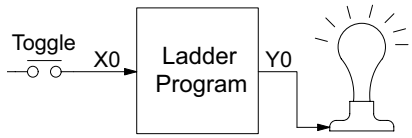
Counter – When using a counter inside a stage, the stage must be active for one scan before the input to the counter makes a 0–1 transition. Otherwise, there is no real transition and the counter will not count. The ordinary Counter instruction does have a restriction inside stages: it may not be reset from other stages using the RST instruction for the counter bit. However, the special Stage Counter provides a solution (see next paragraph).

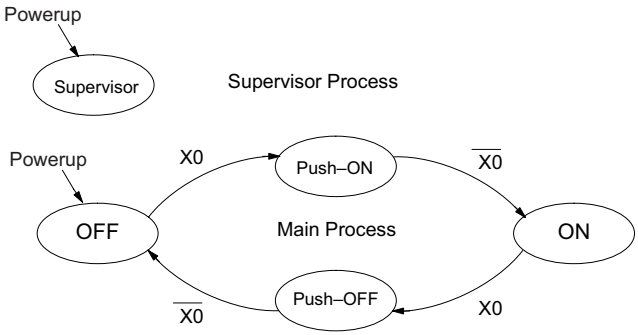
Stage Counter – The Stage Counter has the benefit that its count may be globally reset from other stages by using the RST instruction. It has a count input, but no reset input. This is the only difference from a standard counter instruction.

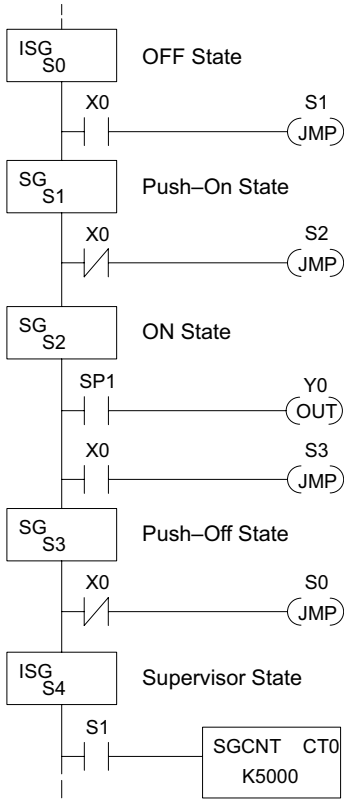
Drum – Realize the drum sequencer is its own process, and is a different programming method than stage programming. If you need to use a drum and stages, be sure to place the drum instruction in an ISG stage that is always active.

Using a Stage as a Supervisory Process

You may recall the light bulb on-off controller example from earlier in this chapter. For the purpose of illustration, suppose we want to monitor the "productivity" of the lamp process by counting the number of on-off cycles that occur. This application will require the addition of a simple counter, but the key decision is in where to put the counter.







New stage programming students will typically try to place the counter inside one of the stages of the process they are trying to monitor. The problem with this approach is that the stage is active only part of the time. In order for the counter to count, the count input must transition from off to on at least one scan after its stage activates. Ensuring this requires extra logic that can be tricky. In this case, we only need to add another supervisory stage as shown above, to "watch" the main process. The counter inside the supervisor stage uses the stage bit S1 of the main process as its count input. *Stage bits used as a contact let us monitor a process!*

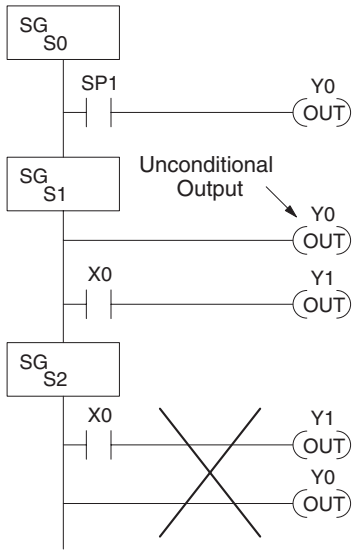
***NOTE**: Both the Supervisor stage and the OFF stage are initial stages. The supervisor stage remains active indefinitely.*

Stage Counter

The counter in the above example is a special Stage Counter. Note that it does not have a reset input. The count is reset by executing a Reset instruction, naming the counter bit (CT0 in this case). The Stage Counter has the benefit that its count may be globally reset from other stages. The standard Counter instruction does not have this global reset capability. You may still use a regular Counter instruction inside a stage, however, the reset input to the counter is the only way to reset it.

Unconditional Outputs

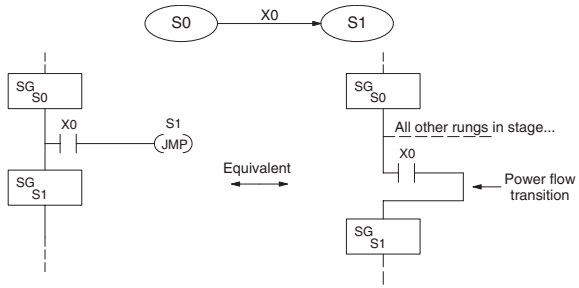
As in most example programs in this chapter and Stage 0 to the right, your application may require a particular output to be ON unconditionally when a particular stage is active. Until now, the examples always use the SP1 special relay contact (always on) in series with the output coils. It's possible to omit the contact, as long as you place any unconditional outputs first (at the top) of a stage section of ladder. The first rung of Stage 1 does this.



WARNING: Unconditional outputs placed elsewhere in a stage do not necessarily remain on when the stage is active. In Stage 2 to the right, Y0 is shown as an unconditional output, but its powerflow comes from the rung above. So, Y0 status will be the same, as Y1 is not correct.

Power Flow Transition Technique

Our discussion of state transitions has shown how the Stage JMP instruction makes the current stage inactive and the next stage (named in the JMP) active. As an alternative way to enter this in ***Direct***SOFT, you may use the power flow method for stage transitions. The main requirement is the current stage be located directly above the next (jump-to) stage in the ladder program. This arrangement is shown in the diagram below, by stages S0 and S1, respectively.



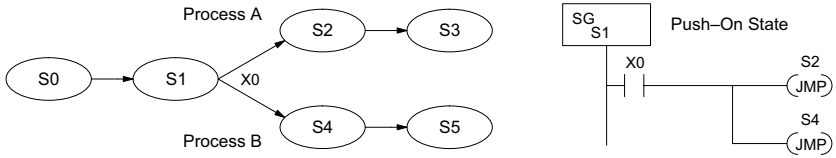
Recall the Stage JMP instruction may occur anywhere in the current stage, and the result is the same. However, power flow transitions (shown above) must occur as the last rung in a stage. All other rungs in the stage will precede it. The power flow transition method is also achievable on the handheld programmer, by simply following the transition condition with the Stage instruction for the next stage.

The power flow transition method does eliminate one Stage JMP instruction, its only advantage. However, it is not as easy to make program changes as using the Stage JMP. Therefore, we advise using Stage JMP transitions for most programs.

Parallel Processing Concepts

Parallel Processes

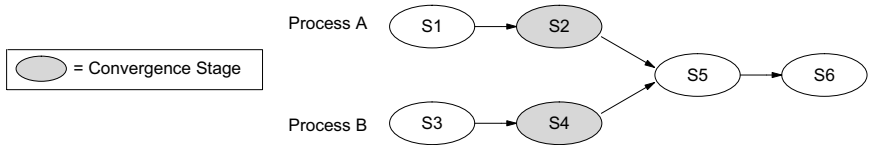
Previously in this chapter we discussed how a state may transition to either one state or another, called an exclusive transition. In other cases, we may need to branch simultaneously to two or more parallel processes, as shown below. It is acceptable to use all JMP instructions as shown, or we could use one JMP and a Set Stage bit instruction(s) (at least one must be a JMP, in order to leave S1). Remember that all instructions in a stage execute, even when it transitions (the JMP is not a GOTO).



Note that if we want Stages S2 and S4 to energize exactly on the same scan, both stages must be located below Stage S1 in the ladder program (see the explanation at the bottom of page 7–6). Overall, parallel branching is easy!

Converging Processes

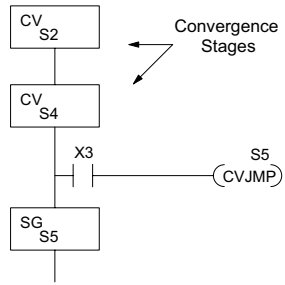
Now we consider the opposite case of parallel branching, which is *converging processes*. This simply means we stop doing multiple things and continue doing one thing at a time. In the figure below, processes A and B converge when stages S2 and S4 transition to S5 at some point in time. So, S2 and S4 are *Convergence Stages.*



Convergence Stages (CV)

230
240
250-1
260
262

While the converging principle is simple enough, it brings a new complication. As parallel processing completes, the multiple processes almost never finish at the same time. In other words, how can we know whether Stage S2 or S4 will finish last? This is an important point, because we have to decide how to transition to Stage S5.

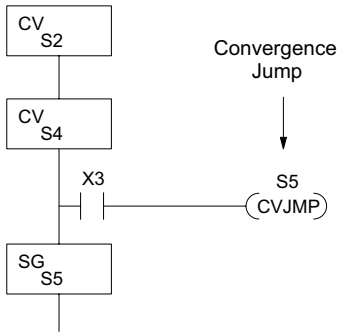


The solution is to coordinate the transition condition out of convergence stages. We accomplish this with a stage type designed for this purpose: the Convergence Stage (type CV). In the example to the right, convergence stages S2 and S4 are required to be grouped as shown. **No logic is permitted between CV stages!** The transition condition (X3 in this case) must be located in the last convergence stage. The transition condition only has power flow when all convergence stages in the group are active.

Convergence Jump (CVJMP)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

Recall the last convergence stage only has power flow when all CV stages in the group are active. To complement the convergence stage, we need a new jump instruction. The Convergence Jump (CVJMP) shown to the right will transition to Stage S5 when X3 is active (as one might expect), but it also *automatically resets all convergence stages in the group*. This makes the CVJMP jump a very powerful instruction. Note that this instruction may only be used with convergence stages.



Convergence Stage Guidelines

The following summarizes the requirements in the use of convergence stages, including some tips for their effective application:

- A convergence stage is to be used as the last stage of a process that is running in parallel to another process or processes. A transition to the convergence stage means that a particular process is through, and represents a waiting point until all other parallel processes also finish.
- The maximum number of convergence stages that make up one group is 17. In other words, a maximum of 17 stages can converge into one stage.
- Convergence stages of the same group must be placed together in the program, connected on the power rail without any other logic in between.
- Within a convergence group, the stages may occur in any order, top to bottom. It does not matter which stage is last in the group, because all convergence stages have to be active before the last stage has power flow.
- The last convergence stage of a group may have ladder logic within the stage. However, this logic will not execute until all convergence stages of the group are active.
- The convergence jump (CVJMP) is the intended method to be used to transition from the convergence group of stages to the next stage. The CVJMP resets all convergence stages of the group, and energizes the stage named in the jump.
- The CVJMP instruction must only be used in a convergence stage, as it is invalid in regular or initial stages.
- Convergence Stages or CVJMP instructions may not be used in subroutines or interrupt routines.

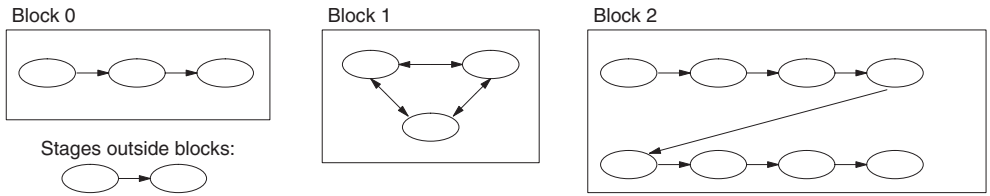
Managing Large Programs

A stage may contain a lot of ladder rungs, or only one or two program rungs. For most applications, good program design will ensure the average number of rungs per stage will be small. However, large application programs will still create a large number of stages. We introduce a new construct that will help us organize related stages into groups called blocks. So, program organization is the main benefit of the use of stage blocks.

Stage Blocks (BLK, BEND)

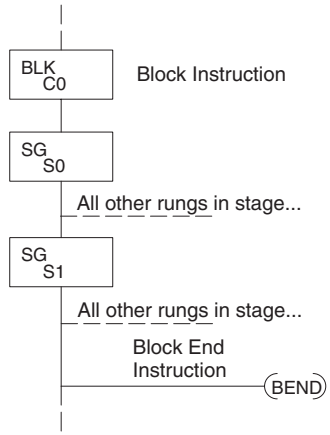
☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

A block is a section of ladder program that contains stages. In the figure below, each block has its own reference number. Like stages, a stage block may be active or inactive. Stages inside a block are not limited in how they may transition from one to another. Note the use of stage blocks does not require each stage in a program to reside inside a block, shown below by the "stages outside blocks."



A program with 20 or more stages may be considered large enough to use block grouping (however, their use is not mandatory). When used, the number of stage blocks should probably be two or higher, because the use of one block provides a negligible advantage.

A block of stages is separated from other ladder logic with special beginning and ending instructions. In the figure to the right, the BLK instruction at the top marks the start of the stage block. At the bottom, the Block End (BEND) marks the end of the block. The stages in between these boundary markers (S0 and S1 in this case) and their associated rungs make up the block.



Note the block instruction has a reference value field (set to "C0" in the example). The block instruction borrows or uses a control relay contact number, so that other parts of the program can control the block. *Any control relay number (such as C0) used in a BLK instruction is not available for use as a control relay.*

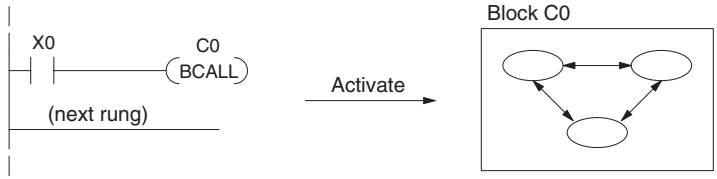
***NOTE**: The stages within a block must be regular stages (SG) or convergence stages (CV), so, they cannot be initial stages. The numbering of stages inside stage blocks can be in any order, and is completely independent from the numbering of the blocks.*

Block Call (BCALL)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

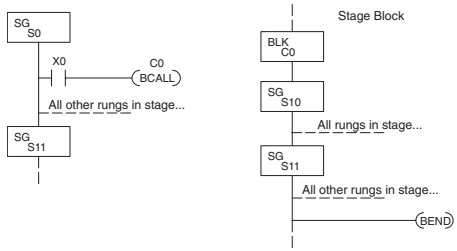
The purpose of the Block Call instruction is to activate a stage block. At powerup or upon Program-to-Run mode transitions, all stage blocks and the stages within them are inactive. Shown in the figure below, the Block Call instruction is a type of output coil. When the X0 contact is closed, the BCALL will cause the stage block referenced in the instruction (C0) to become active. When the BCALL is turned off, the corresponding stage block and the stages within it become inactive.

We must avoid confusing block call operation with how a "subroutine call" works. After a BCALL coil executes, program execution continues with the next program rung. Whenever program execution arrives at the ladder location of the stage block named in the BCALL, then logic within the block executes because the block is now active. Similarly, do not classify the BCALL as type of state transition (is not a JMP).



When a stage block becomes active, the first stage in the block automatically becomes active on the same scan. The "first" stage in a block is the one located immediately under the block (BLK) instruction in the ladder program. So, that stage plays a similar role to the initial type stage we discussed earlier.

The Block Call instruction may be used in several contexts. Obviously, the first execution of a BCALL must occur outside a stage block, since stage blocks are initially inactive. Still, the BCALL may occur on an ordinary ladder rung, or it may occur within an active stage as shown below. Note that either turning off the BCALL or turning off the stage containing the BCALL will deactivate the corresponding stage block. You may also control a stage block with a BCALL in another stage block.



***NOTE**: Stage Block may come before or after the location of the BCALL instruction in the program.*

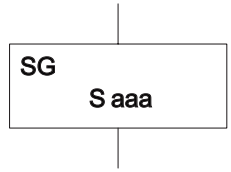
The BCALL may be used in many ways or contexts, so it can be difficult to find the best usage. Remember the purpose of stage blocks is to help you organize the application problem by grouping related stages. Remember that initial stages must exist *outside* stage blocks.

RLL*PLUS* (Stage) Instructions

Stage (SG)

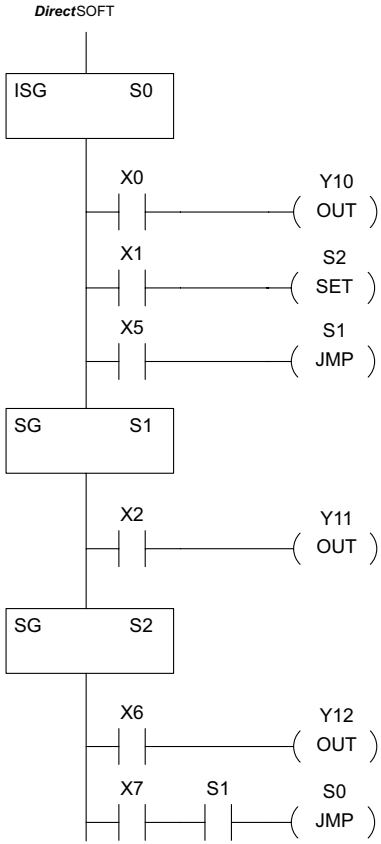
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Stage instructions are used to create structured RLL*PLUS* programs. Stages are program segments that can be activated by transitional logic, a Jump or a Set Stage that is executed from an active stage. Stages are deactivated one scan after transitional logic, a Jump, or a Reset Stage instruction is executed.



Operand Data Type		D2-230 Range	D2-240 Range	D2-250-1 Range	D2-260/D2-262 Range
		aaa	aaa	aaa	aaa
Stage	S	0-377	0-777	0-1777	0-1777

The following example is a simple RLL*PLUS* program. This program utilizes the Initial Stage, as well as Stage and Jump instructions to create a structured program.



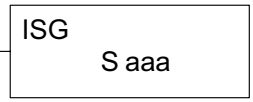
Handheld Programmer Keystrokes

ISG	→	S(SG)	0	ENT	
STR	→	X(IN)	0	ENT	
OUT	→	Y(OUT)	1	0	ENT
STR	→	X(IN)	1	ENT	
SET	→	S(SG)	2	ENT	
STR	→	X(IN)	5	ENT	
JMP	→	S(SG)	1	ENT	
SG	→	S(SG)	1	ENT	
STR	→	X(IN)	2	ENT	
OUT	→	Y(OUT)	1	1	ENT
SG	→	S(SG)	2	ENT	
STR	→	X(IN)	6	ENT	
OUT	→	Y(OUT)	1	2	ENT
STR	→	X(IN)	7	ENT	
AND	→	S(SG)	1	ENT	
JMP	→	S(SG)	0	ENT	

Initial Stage (ISG)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Initial Stage instruction is normally used as the first segment of an RLL*PLUS* program. Initial stages will be active when the CPU enters the run mode allowing for a starting point in the program. Initial Stages are also activated by transitional logic, a jump or a set stage executed from an active stage. Initial Stages are deactivated one scan after transitional logic, a jump, or a reset stage instruction is executed. Multiple Initial Stages are allowed in a program.



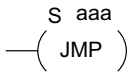
Operand Data Type		D2-230 Range	D2-240 Range	D2-250-1 Range	D2-260/D2-262 Range
		aaa	aaa	aaa	aaa
Stage	S	0-377	0-777	0-1777	0-1777

NOTE: *If the ISG is within the retentive range for stages, the ISG will remain in the state it was in before power down and will NOT turn itself on during the first scan.*

Jump (JMP)

☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Jump instruction allows the program to transition from an active stage which contains the jump instruction to another stage which is specified in the instruction. The jump will occur when the input logic is true. The active stage that contains the Jump will be deactivated one scan after the Jump instruction is executed.

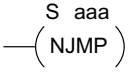


Operand Data Type		D2-230 Range	D2-240 Range	D2-250-1 Range	D2-260/D2-262 Range
		aaa	aaa	aaa	aaa
Stage	S	0-377	0-777	0-1777	0-1777

Not Jump (NJMP)

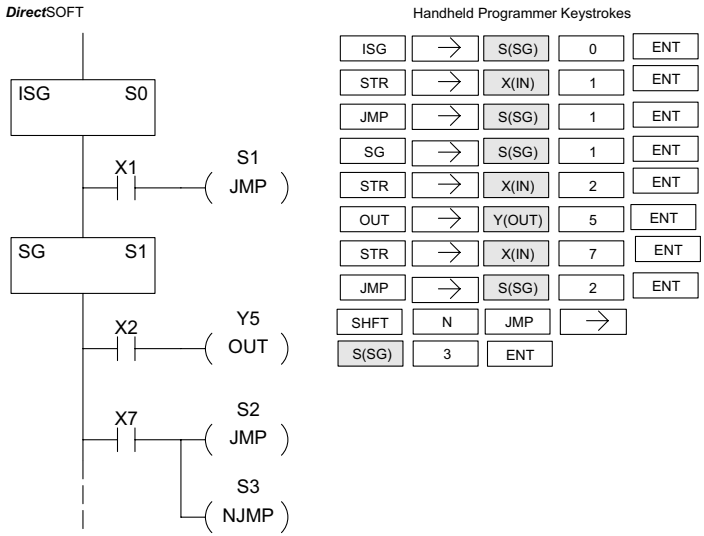
☑ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Not Jump instruction allows the program to transition from an active stage which contains the jump instruction to another which is specified in the instruction. The jump will occur when the input logic is off. The active stage that contains the Not Jump will be deactivated one scan after the Not Jump instruction is executed.



Operand Data Type		D2-230 Range	D2-240 Range	D2-250-1 Range	D2-260/D2-262 Range
		aaa	aaa	aaa	aaa
Stage	S	0-377	0-777	0-1777	0-1777

In the following example, when the CPU begins program execution, only ISG 0 will be active. When X1 is on, the program execution will jump from Initial Stage 0 to Stage 1. In Stage 1, if X2 is on, output Y5 will be turned on. If X7 is on, program execution will jump from Stage 1 to Stage 2. If X7 is off, program execution will jump from Stage 1 to Stage 3.



Converge Stage (CV) and Converge Jump (CVJMP)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Converge Stage instruction is used to group certain stages by defining them as Converge Stages.

CV
S aaa

When all of the Converge Stages within a group become active, the CVJMP instruction (and any additional logic in the final CV stage) will be executed. All preceding CV stages *must* be active before the final CV stage logic can be executed. All Converge Stages are deactivated one scan after the CVJMP instruction is executed.

Additional logic instructions are only allowed following the last Converge Stage instruction and before the CVJMP instruction. Multiple CVJUMP instructions are allowed.

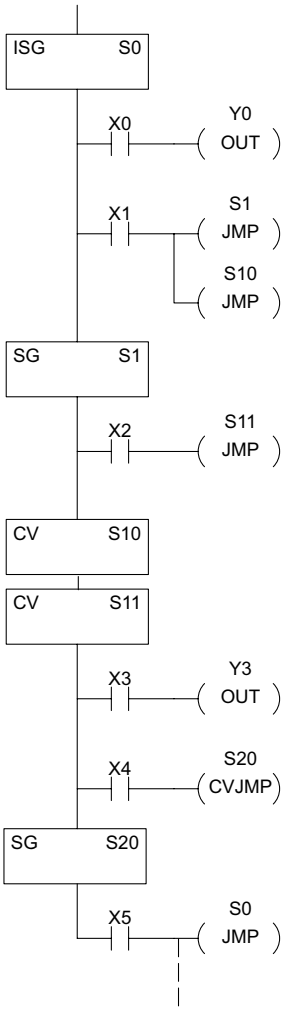
S aaa
(CVJMP)

Converge Stages must be programmed in the main body of the application program. This means they cannot be programmed in Subroutines or Interrupt Routines.

Operand Data Type		D2-240 Range	D2-250-1 Range	D2-260/D2-262 Range
		aaa	aaa	aaa
Stage	S	0-777	0-1777	0-1777

In the following example, when Converge Stages S10 and S11 are *both* active, the CVJMP instruction will be executed when X4 is on. The CVJMP will deactivate S10 and S11, and activate S20. Then, if X5 is on, the program execution will jump back to the initial stage, S0.

DirectSOFT



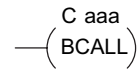
Handheld Programmer Keystrokes

ISG	→	S(SG)	0	ENT				
STR	→	X(IN)	0	ENT				
OUT	→	Y(OUT)	0	ENT				
STR	→	X(IN)	1	ENT				
JMP	→	S(SG)	1	ENT				
JMP	→	S(SG)	1	0	ENT			
SG	→	S(SG)	1	ENT				
STR	→	X(IN)	2	ENT				
JMP	→	S(SG)	1	1	ENT			
SHFT	C	V	→	S(SG)	1	0	ENT	
SHFT	C	V	→	S(SG)	1	1	ENT	
STR	→	X(IN)	3	ENT				
OUT	→	Y(OUT)	3	ENT				
STR	→	X(IN)	4	ENT				
SHFT	C	V	SHFT	JMP	S(SG)	2	0	ENT
SG	→	S(SG)	2	0	ENT			
STR	→	X(IN)	5	ENT				
JMP	→	S(SG)	0	ENT				

Block Call (BCALL)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The stage block instructions are used to activate a block of stages. The Block Call, Block, and Block End instructions must be used together. The BCALL instruction is used to activate a stage block. There are several things you need to know about the BCALL instruction.



Uses CR Numbers — The BCALL appears as an output coil, but does not actually refer to a Stage number as you might think. Instead, the block is identified with a Control Relay (Caaa). This control relay cannot be used as an output anywhere else in the program.

Must Remain Active — The BCALL instruction actually controls all the stages between the BLK and the BEND instructions even after the stages inside the block have started executing. The BCALL must *remain active* or all the stages in the block will be turned off automatically. *If either the BCALL instruction, or the stage that contains the BCALL instruction goes off, then the stages in the defined block will be turned off automatically.*

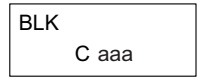
Activates First Block Stage — When the BCALL is executed it automatically activates the first stage following the BLK instructions.

Operand Data Type	D2-240 Range	D2-250-1 Range	D2-260/D2-262 Range
	aaa	aaa	aaa
Control Relay C	0-777	0-1777	0-3777

Block (BLK)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The Block instruction is a label which marks the beginning of a block of stages that can be activated as a group. A Stage instruction must immediately follow the Start Block instruction. Initial Stage instructions are not allowed in a block. The control relay (Caaa) specified in Block instruction must not be used as an output anywhere else in the program.



Block End (BEND)

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

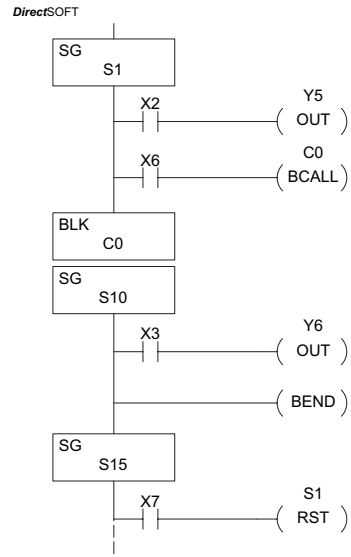
The Block End instruction is a label used with the Block instruction. It marks the end of a block of stages. There is no operand with this instruction. Only one Block End is allowed per Block Call.

Operand Data Type	D2-240 Range	D2-250-1 Range	D2-260/D2-262 Range
	aaa	aaa	aaa
Control Relay C	0-777	0-1777	0-3777

In this example, the Block Call is executed when stage 1 is active and X6 is on. The Block Call then automatically activates stage S10, which immediately follows the Block instruction.

This allows the stages between S10 and the Block End instruction to operate as programmed. If the BCALL instruction is turned off, or if the stage containing the BCALL instruction is turned off, then all stages between the BLK and BEND instructions are automatically turned off.

If you examine S15, you will notice that X7 could reset Stage S1, which would disable the BCALL, thus resetting all stages within the block.

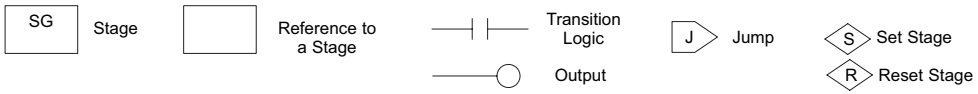


Handheld Programmer Keystrokes

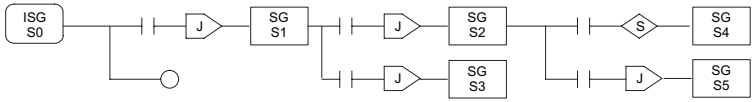
SG	→	S(SG)	1	ENT					
STR	→	X(IN)	2	ENT					
OUT	→	Y(OUT)	5	ENT					
STR	→	X(IN)	6	ENT					
SHFT	B	C	A	L	L	→	C(CR)	0	ENT
SHFT	B	L	K	→	C(CR)	0	ENT		
SG	→	S(SG)	1	0	ENT				
STR	→	X(IN)	3	ENT					
OUT	→	Y(OUT)	6	ENT					
SHFT	B	E	N	D	ENT				
SG	→	S(SG)	1	5	ENT				
STR	→	X(IN)	7	ENT					
RST	→	S(SG)	1	ENT					

Stage View in DirectSOFT

The Stage View option in *Direct*SOFT will let you view the ladder program as a flow chart. The figure below shows the symbol convention used in the diagrams. You may find the stage view useful as a tool to verify that your stage program has faithfully reproduced the logic of the state transition diagram you intend to realize.



The following diagram is a typical stage view of a ladder program containing stages. Note the left-to-right direction of the flow chart.



Questions and Answers about Stage Programming

We include the following commonly-asked questions about Stage Programming as an aid to new students. All question topics are covered in more detail in this chapter.

Q. What does stage programming do that I can't do with regular RLL programs?

A. Stages allow you to identify all the states of your process before you begin programming. This approach is more organized, because you divide a ladder program into sections. As stages, these program sections are active only when they are actually needed by the process. Most processes can be organized into a sequence of stages, connected by event-based transitions.

Q. Isn't a stage really like a software subroutine?

A. No, it is very different. A subroutine is called by a main program when needed, and executes only once before returning to the point from which it was called. A stage, however, is part of the main program. It represents a state of the process, so an active stage executes on every scan of the CPU until it becomes inactive.

Q. What are Stage Bits?

A. A stage bit is a single bit in the CPU's image register, representing the active/inactive status of the stage in real time. For example, the bit for Stage 0 is referenced as "S0". If S0 = 0, then the ladder rungs in Stage 0 are bypassed (not executed) on each CPU scan. If S0 = 1, then the ladder rungs in Stage 0 are executed on each CPU scan. Stage bits, when used as contacts, allow one part of your program to monitor another part by detecting stage active/inactive status.

Q. How does a stage become active?

A. There are three ways:

- If the Stage is an initial stage (ISG), it is automatically active at powerup.
- Another stage can execute a Stage JMP instruction naming this stage, which makes it active upon its next occurrence in the program.
- A program rung can execute a Set Stage Bit instruction (such as SET S0).

Q. How does a stage become inactive?

A. There are three ways:

- Standard Stages (SG) are automatically inactive at powerup.
- A stage can execute a Stage JMP instruction, resetting its Stage Bit to 0.
- Any rung in the program can execute a Reset Stage Bit instruction (such as RST S0).

Q. What about the power flow technique of stage transitions?

A. The power flow method of connecting adjacent stages (directly above or below in the program) actually is the same as the Stage Jump instruction executed in the stage above, naming the stage below. Power flow transitions are more difficult to edit in *Direct*SOFT; we list them separately from two preceding questions.

Q. Can I have a stage that is active for only one scan?

A. Yes, but this is not the intended use for a stage. Instead, make a ladder rung active for one scan by including a stage Jump instruction at the bottom of the rung. Then the ladder will execute on the last scan before its stage jumps to a new one.

Q. Isn't a Stage JMP just like a regular GOTO instruction used in software?

A. No, it is very different. A GOTO instruction sends the program execution immediately to the code location named by the GOTO. A Stage JMP simply resets the Stage Bit of the current stage, while setting the Stage Bit of the stage named in the JMP instruction. Stage bits are 0 or 1, determining the inactive/active status of the corresponding stages. A stage JMP has the following results:

- When the JMP is executed, the remainder of the current stage's rungs are executed, even if they reside past (under) the JMP instruction. On the following scan, that stage is not executed, because it is inactive.
- The Stage named in the Stage JMP instruction will be executed upon its next occurrence. If located past (under) the current stage, it will be executed on the same scan. If located before (above) the current stage, it will be executed on the following scan.

Q. How can I know when to use stage JMP, versus a Set Stage Bit or Reset Stage Bit?

A. These instructions are used according to the state diagram topology you have derived:

- Use a Stage JMP instruction for a state transition... moving from one state to another.
- Use a Set Stage Bit instruction when the current state is spawning a new parallel state or stage sequence, or when a supervisory state is starting a state sequence under its command.
- Use a Reset Bit instruction when the current state is the last state in a sequence and its task is complete, or when a supervisory state is ending a state sequence under its command.

Q. What is an initial stage, and when do I use it?

A. An initial stage (ISG) is automatically active at powerup. Afterwards, it works just like any other stage. You can have multiple initial stages, if required. Use an initial stage for a ladder that must always be active, or as a starting point.

Q. Can I have place program ladder rungs outside of the stages, so they are always on?

A. It is possible, but it's not good software design practice. Place a ladder that must always be active in an initial stage, and do not reset that stage or use a Stage JMP instruction inside it. It can start other stage sequences at the proper time by setting the appropriate Stage Bit(s).

Q. Can I have more than one active stage at a time?

A. Yes, and this is a normal occurrence for many programs. However, it is important to organize your application into separate processes, each made up of stages. And a good process design will be mostly sequential, with only one stage on at a time. However, all the processes in the program may be active simultaneously.

PID Loop Operation (D2-250-1/ D2-260/D262)

In This Chapter...

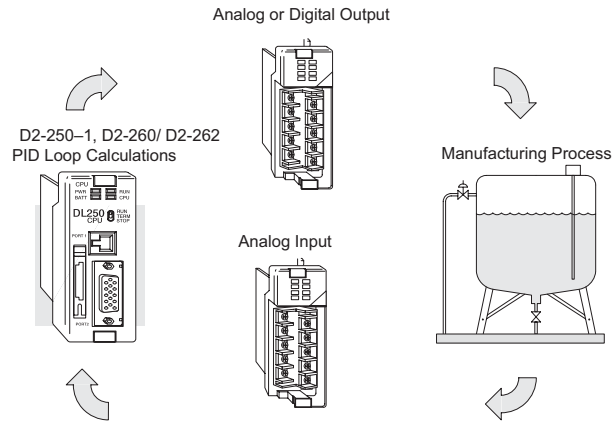
D2-250-1, D2-260 and D2-262 PID Loop Features

Main Features

The D2-250-1 D2-260 and D2-262 CPUs process loop control offers a sophisticated set of feature to address many application needs. The main features are:

- D2-250-1 – up to 4 loops, individual programmable sample rates
- D2-260 and D2-262 – up to 16 loops, individual programmable sample rates
- Manual, Automatic and Cascade loop operation modes
- Two types of bumpless transfer available
- Full-featured alarms
- Ramp/soak generator with up to 16 segments
- Auto Tuning

The D2-250–1, D2-260 and D2-262 CPUs have process control loop capability in addition to ladder program execution. You can select and configure up to four loops for the D2-250-1 or sixteen loops for the D2-260 and D2-262. All sensor and actuator wiring connects to standard DL205 I/O modules, as shown below. All process variables, gain values, alarm levels, etc., associated with each loop reside in a Loop Variable Table in the CPU. The CPU reads process variable (PV) inputs during each scan. Then, it makes PID loop calculations during a dedicated time slice on each PLC scan, updating the control output value. The control loops use a Proportional-Integral-Derivative (PID) algorithm to generate the control output. This chapter describes how the loops operate, and how to configure and tune the loops.



*Direct*SOFT programming software is used for configuring analog control loops in the DL205. *Direct*SOFT uses dialog boxes to help you set up the individual loops. After completing the setup, you can use *Direct*SOFT's PID Trend View to tune each loop. The PID starting address (V7640) and the number of loops (V7641) are stored in Flash. The PID configuration and tuning parameters are stored in V-memory (battery-backed RAM), which the user should consider setting as retentive. The loop parameters also may be saved to disk for recall later.

PID Loop Feature	Specifications
Number of loops	D2-250-1 - selectable up to 4; D2-260 and D2-262 - selectable up to 16
CPU V-memory needed	32 words (V locations) per loop selected, 64 words if using ramp/soak
PID algorithm	Position or Velocity form of the PID equation
Control Output polarity	Selectable direct-acting or reverse-acting
Error term curves	Selectable as linear, square root of error, and error squared
Loop update rate (time between PID calculation)	0.05 to 99.99 seconds, user programmable
Minimum loop update rate	0.05 seconds for 1 to 4 loops (D2-250-1, D2-260 and D2-262) 0.1 seconds for 5 to 8 loops (D2-260 and D2-262) 0.2 seconds for 9 to 16 loops (D2-260 and D2-262)
Loop modes	Automatic, Manual (operator control), or Cascade control
Ramp/Soak Generator	Up to 8 ramp/soak steps (16 segments) per loop with indication of ramp/soak step number
PV curves	Select standard linear, or square-root extract (for flow meter input)
Set Point Limits	Specify minimum and maximum setpoint values
Process Variable Limits	Specify minimum and maximum Process Variable values
Proportional Gain	Specify gains of 0.01 to 99.99
Integrator (Reset)	Specify reset time of 0.1 to 99.99 in units of seconds or minutes
Derivative (Rate)	Specify the derivative time from 0.01 to 99.99 seconds
Rate Limits	Specify derivative gain limiting from 1 to 20
Bumpless Transfer I	Automatically sets the bias equal to the control output and the setpoint equal to the process variable when control switches from manual to automatic.
Bumpless Transfer II	Automatically sets the bias equal to the control output when control switches from manual to automatic.
Step Bias	Provides proportional bias adjustment for large setpoint changes
Anti-windup (Freeze Bias)	For position form of PID, this inhibits integrator action when the control output reaches 0% or 100 % (speeds up loop recovery when output recovers from saturation)
Error Deadband	Specify a tolerance (plus and minus) for the error term (SP–PV), so that no change in control output value is made

Alarm Feature	Specifications
PV Alarm Hysteresis	Specify 1 to 200 (word/binary) does not affect all alarms, such as PV Rate-of-Change Alarm
PV Alarm Points	Select PV alarm settings for Low–Low, Low, High, and High-High conditions
PV Deviation	Specify alarms for two ranges of PV deviation from the setpoint value
Rate of Change	Detect when PV exceeds a rate of change limit you specify

Introduction to PID Control

What is PID Control?

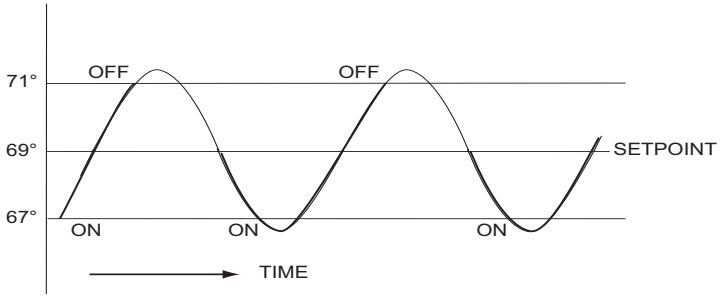
In this discussion, we will explain why PID control is used in process control instead of trying to provide control by simply using an analog input and a discrete output.

There are many types of analog controllers available, and the proper selection will depend upon the particular application. There are two types of analog controllers that are used throughout industry:

1. The ON-OFF controller, sometimes referred to as an open loop controller.
2. The PID controller, sometimes called a closed loop controller.

Regardless of type, analog controllers require input signals from electronic sensors such as pressure, differential pressure, level, flow meter or thermocouples. As an example, one of the most common analog control applications is located in your house for controlling either heat or air conditioning, the thermostat.

You wish for your house to be at a comfortable temperature so you set a thermostat to a desired temperature (setpoint). You then select the comfort mode, either heat or A/C. A temperature sensing device, normally a thermistor, is located within the thermostat. If the thermostat is set for heat and the setpoint is set for 69°F, the furnace will be turned on to provide heat at, normally, 2°F below setpoint. In this case, it would turn on at 67°F. When the temperature reaches 71°F, 2°F above setpoint, the furnace will turn off. In the opposite example, if the thermostat is set for A/C (cooling), the thermostat will turn the A/C unit on/off opposite the heat setting. For instance, if the thermostat is set to cool at 76°F, the A/C unit will turn on when the sensed temperature reaches 2°F above setpoint or 78°F, and turn off when the temperature reaches 74°F. This would be considered to be an ON-OFF controller. The waveform below shows the action of the heating cycle. Note that there is a slight overshoot at the turn-off point, also a slight undershoot at the turn-on point.



The ON-OFF controller is used in some industrial control applications, but is not practical in the majority of industrial control processes.

The most common process controller that is used in industry is the PID controller.

The PID controller controls a continuous feedback loop that keeps the process output (control variable) flowing normally by taking corrective action whenever there is a deviation from the desired value (setpoint) of the process variable (PV) such as, rate of flow, temperature, voltage, etc. An *error* occurs when an operator manually changes the setpoint or when an event (valve opened, closed, etc.) or a disturbance (cold water, wind, etc.) changes the load, thus causing a change in the process variable.

The PID controller receives signals from sensors and computes corrective action to the actuator from a computation based on the error (Proportional), the sum of all previous errors (Integral) and the rate of change of the error (Derivative).

We can look at the PID controller in more simple terms. Take the cruise control on an automobile as an example. Let's say that we are cruising on an interstate highway in a car equipped with cruise control. The driver decides to engage the cruise control by turning it ON, then he manually brings the car to the desired cruising speed, say 70 miles per hour. Once the cruise speed is reached, the SET button is pushed fixing the speed at 70 mph, the setpoint. Now, the car is cruising at a steady 70 mph until it comes to a hill to go up. As the car goes up the hill, it tends to slow down. The speed sensor senses this and causes the throttle to increase the fuel to the engine. The vehicle speeds up to maintain 70 mph without jerking the car and it reaches the top at the set speed. When the car levels out after reaching the top of the hill it will speed up. The speed sensor senses this and signals the throttle to provide less fuel to the engine, thus, the engine slows down allowing the car to maintain the 70 mph speed. How does this application apply to PID control? Lets look at the function of P, I and D terms:

- Proportional - is commonly referred to as Proportional Gain. The proportional term is the corrective action which is proportional to the error, that is, the change of the manipulated variable is equal to the proportional gain multiplied by the error (the activating signal). In mathematical terms:

  Proportional action = proportional gain X error
  Error = Setpoint (SP) - Process Variable (PV)

- Applying this to the cruise control, the speed was set at 70 mph which is the Setpoint. The speed sensor senses the actual speed of the car and sends this signal to the cruise controller as the Process Variable (PV). When the car is on a level highway, the speed is maintained at 70 mph, thus, no error since the error would be SP - PV = 0. When the car goes up the hill, the speed sensor detected a slow down of the car, SP-PV = error. The proportional gain would cause the output of the speed controller to bring the car back to the setpoint of 70 mph. This would be the Controlled Output.

- Integral - this term is often referred to as Reset action. It provides additional compensation to the control output, which causes a change in proportion to the value of the error over a period of time. In other words, the reset term is the integral sum of the error values over a period of time.

- Derivative - this term is referred to as rate. The Rate action adds compensation to the control output, which causes a change in proportion to the rate of change of error. Its job is to anticipate the probable growth of the error and generate a contribution to the output in advance.

Introducing DL205 PID Control

The DL205 is capable of controlling a process variable such as those already mentioned. As previously mentioned, the control of a variable, such as temperature, at a given level (setpoint) as long as there are no disturbances (cold water) in the process.

The DL205 PLC has the ability to directly accept signals from electronic sensors, such as thermocouples, pressure, VFDs, etc. These signals may be used in mathematically derived control systems.

In addition, the DL205 has built-in PID control algorithms that can be implemented. The basic function of PID closed loop process control is to maintain certain process characteristics at desired setpoints. As a rule, the process deviates from the desired setpoint reference as a result of load material changes and interaction with other processes. During this control, the actual condition of the process characteristics (liquid level, temperature, motor control, etc.) is measured as a *process variable* (PV) and compared with the target setpoint (SP). When deviations occur, an error is generated by the difference between the process variable (actual value) and the setpoint (desired value). Once an error is detected, the function of the control loop is to modify the control output in order to force the error to zero.

The DL205 PID control provides feedback loops using the PID algorithm. The control output is computed from the measured process variable as follows:

Let:

- K_c = proportional gain
- T_i = Reset or integral time
- T_d = Derivative time or rate
- SP = Setpoint
- PV(t) = Process Variable at time "t"
- e(t) = SP-PV(t) = PV deviation from setpoint at time "t" or PV error.

Then:

- M(t) = Control output at time "t"

$$M(t) = Kc \left[e(t) + 1/T_i \int_0^t e(x)\,dx + T_d\, d/dt\, e(t) \right] + M_o$$

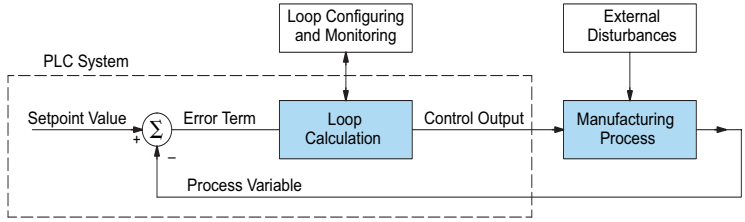
The analog input module receives the process variable in analog form along with an operator entered setpoint; the CPU computes the error. The error is used in the algorithm computation to provide corrective action at the control output. The function of the control action is based on an output control, which is proportional to the instantaneous error value. The *integral control action* (reset action) provides additional compensation to the control output, which causes a change in proportion to the value of the change of error over a period of time. The *derivative control action* (rate change) adds compensation to the control output, which causes a change in proportion to the rate of change of error. These three modes are used to provide the desired control action in Proportional (P), Proportional-Integral (PI), or Proportional-Integral-Derivative (PID) control fashion.

Standard DL205 analog input modules are used to interface to field transmitters to obtain the PV. These transmitters normally provide a 4–20 mA current or an analog voltage of various ranges for the control loop.

For temperature control, thermocouple or RTD can be connected directly to the appropriate module. The PID control algorithm, residing in the CPU memory, receives information from the user program, primarily control parameters and setpoints. Once the CPU makes the PID calculation, the result may be used to directly control an actuator connected to a 4–20 mA current output module to control a valve.

With ***Direct***SOFT, additional ladder logic programming, both time proportioning (e.g., heaters for temperature control) and position actuator (e.g., reversible motor on a valve) type of control schemes can be easily implemented. This chapter will explain how to set up the PID control loop, how to implement the software and how to tune the loop.

The following block diagram shows the key parts of a PID control loop. The path from the PLC to the manufacturing process and back to the PLC is the closed loop control.



Process Control Definitions

Manufacturing Process – The set of actions that adds value to raw materials. The process can involve physical changes and/or chemical changes to the material. The changes render the material more useful for a particular purpose, ultimately used in a final product.

Process Variable – The controlled variable part of the process that you wish to control. It may be temperature, pressure, level, flow, composition, density, the ratio of two streams, etc. Also known as the actual value.

Setpoint – The target for the process variable. When all conditions of the process are correct, the process variable will equal the setpoint.

Control Output – The result of the loop calculation, which becomes a command for the process (such as the heater level in an oven). This is sometimes referred to as control variable.

Error Term – The algebraic difference between the process variable and the setpoint. This is the control loop error, and is equal to zero when the process variable is equal to the setpoint (desired) value. A well-behaved control loop is able to maintain a small error term magnitude.

Manipulated Variable – This is used to effect the controlled variable. For example, the fuel used in a furnace might be manipulated in order to control the temperature.

Disturbance – Something in the system that changes such that corrective action is required. For instance, when controlling a flow and the upstream pressure drops, the control valve must open wider in order to keep flow constant. The drop in upstream pressure is the disturbance.

Final Control Element – The physical device used to control the manipulated variable. Valves are probably the most widely used final control element.

Lag Time – The time it takes for the process to respond to a change in manipulated variable. This is also known as the capacitance of the system. When you're in the shower and you turn up the hot water a little, the time it takes before the water gets hot is the lag time.

Dead Time – The time it takes for a change in the process to be recognized. Composition analyzers and quality control are usually sources of significant dead time.

Loop Configuring – Operator-initiated selections which set up and optimize the performance of a control loop. The loop calculation function uses the configuration parameters in real time to adjust gains, offsets, etc.

Loop Monitoring – The function which allows an operator to observe the status and performance of a control loop. This is used in conjunction with the loop configuring to optimize the performance of a loop (minimize the error term).

PID Loop Operation

The Proportional–Integral–Derivative (PID) algorithm is widely used in process control. The PID method of control adapts well to electronic solutions, whether implemented in analog or digital (CPU) components. The DL205 CPU implements the PID equations digitally by solving the basic equations in software. I/O modules serve only to convert electronic signals into digital form (or vice versa).

The DL205 uses two types of PID controls: "position" and "velocity." These terms usually refer to motion control situations, but here we use them in a different sense:

- PID *Position* Algorithm – The control output is calculated so it responds to the displacement (position) of the PV from the SP (error term).
- PID *Velocity* Algorithm – The control output is calculated to represent the rate of change (velocity) for the PV to become equal to the SP.

Position Form of the PID Equation

Referring to the control output equation on page 8-6, the DL205 system CPU approximates the output M(t) using a discrete position form of the PID equation.

- **Let:**

  Ts = Sample rate
  Kc = Proportional gain
  Ki = Kc * (Ts/Ti) = Coefficient of integral term
  Kr = Kc * (Td/Ts) = Coefficient of derivative term
  Ti = Reset or integral time
  Td = Derivative time or rate
  SP = Setpoint
  PV_n = Process variable at n^{th} sample
  en = SP – PV_n = Error at n^{th} sample
  M_o = Value to which the controller output has been initialized

- **Then:**

  M_n = Control output at n^{th} sample

$$M_n = Kc * e_n + Ki \sum_{i=1}^{n} e_i + Kr\,(e_n - e_{n-1}) + M_o$$

This form of the PID equation is referred to as the position form since the actual actuator position is computed. The velocity form of the PID equation computes the change in actuator position. The CPU modifies the standard equation slightly to use the derivative of the process variable instead of the error, as follows:

$$M_n = Kc * e_n + Ki \sum_{i=1}^{n} e_i + Kr\,(PV_n - PV_{n-1}) + M_o$$

These two forms are equivalent unless the setpoint is changed. In the original equation, a large step change in the setpoint will cause a correspondingly large change in the error, resulting in a bump to the process due to derivative action. This bump is not present in the second form of the equation.

The DL205 also combines the integral sum and the initial output into a single term called the bias (Mx). This results in the following set of equations:

$$Mx_o = M_o$$
$$Mx = Ki * e_n + Mx_{n-1}$$
$$M_n = Kc * e_n - Kr(PV_n - PV_{n-1}) + Mx_n$$

The DL205 by default will keep the normalized output M in the range of 0.0 to 1.0. This is done by clamping M to the nearer of 0.0 or 1.0 whenever the calculated output falls outside this range. The DL205 also allows you to specify the minimum and maximum output limit values (within the range 0 to 4095 in BCD if using 12 bit unipolar).

NOTE: *The equations and algorithms, or parts of, in this chapter, are only for references. Analysis of these equations can be found in most good textbooks about process control.*

Reset Windup Protection

Reset windup can occur if reset action (integral term) is specified, and the computation of the bias term Mx is:

$$Mx = Ki * e_n + Mx_{n-1}$$

For example, assume the output is controlling a valve and the PV remains at some value greater than the setpoint. The negative error (e_n) will cause the bias term (Mx) to constantly decrease until the output M goes to 0 closing the valve. However, since the error term is still negative, the bias will continue to decrease, becoming ever more negative. When the PV finally does come back down below the SP, the valve will stay closed until the error is positive for long enough to cause the bias to become positive again. This will cause the process variable to undershoot.

One way to solve the problem is to simply clamp the normalized bias between 0.0 and 1.0. The DL205 CPU does this. However, if this is the only thing that is done, then the output will not move off 0.0 (thus opening the valve) until the PV has become less than the SP. This will also cause the process variable to undershoot.

The DL205 CPU is programmed to solve the overshoot problem by either freezing or adjusting the bias term.

Freeze Bias

If the "Freeze Bias" option is selected when setting up the PID loop (discussed later), then the CPU simply stops changing the bias (Mx) whenever the computed normalized output (M) goes outside the interval 0.0 to 1.0.

$$Mx = Ki * e_n + Mx_{n-1}$$

$$M = Kc * e_n - Kr(PV_n - PV_{n-1}) + Mx$$

$$M_n = 0 \quad \text{if } M < 0$$
$$M_n = M \quad \text{if } 0 \leq M \leq 1$$
$$M_n = 1 \quad \text{if } M > 1$$

$$Mx_n = Mx \quad \text{if } 0 \leq M \leq 1$$
$$Mx_n = Mx_{n-1} \quad \text{otherwise}$$

Thus in this example, the bias will probably not go all the way to zero so that when the PV does begin to come down, the loop will begin to open the valve sooner than it would have if the bias had been allowed to go all the way to zero. This action has the effect of reducing the amount of overshoot.

Adjusting the Bias

The normal action of the CPU is to adjust the bias term when the output goes out of range as shown below.

$$Mx = Ki * e_n + Mx_{n-1}$$
$$M = Kc * e_n - Kr(PV_n - PV_{n-1}) + Mx$$

$$M_n = 0 \quad \text{if } M < 0$$
$$M_n = M \quad \text{if } 0 \leq M \leq 1$$
$$M_n = 1 \quad \text{if } M > 1$$

$$Mx_n = Mx \quad \text{if } 0 \leq M \leq 1$$
$$Mx_n = M_n - Kc * e_n - Kr(PV_n - PV_{n-1}) \quad \text{otherwise}$$

By adjusting the bias, the valve will begin to open as soon as the PV begins to come down. If the loop is properly tuned, overshoot can be eliminated entirely. If the output went out of range due to a setpoint change, then the loop probably will oscillate, because we must wait for the bias term to stabilize again.

The choice of whether to use the default loop action or to freeze the bias is dependent on the application. **If large, step changes to the setpoint are anticipated; then it is probably better to select the freeze bias option** (see page 8-34).

Step Bias Proportional to Step Change in SP

This feature reduces oscillation caused by a step change in setpoint when the adjusting bias feature is used.

$Mx = Mx * SP_n / SP_{n-1}$ if the loop is direct acting

$Mx = Mx * SP_{n-1} / SP_n$ if the loop is reverse acting

$Mx_n = 0$ if $Mx < 0$

$Mx_n = Mx$ if $0 \leq Mx \leq 1$

$Mx_n = 1$ if $M > 1$

Eliminating Proportional, Integral or Derivative Action

It is not always necessary to run a full-three mode PID control loop. Most loops require only the PI terms or just the P term. Parts of the PID equation may be eliminated by choosing appropriate values for the gain (Kc), reset (Ti) and rate (Td) yielding a P, PI, PD, I and even an ID and a D loop.

Eliminating Integral Action The effect of integral action on the output may be eliminated by setting Ti = 9999. When this is done, the user may then manually control the bias term (Mx) to eliminate any steady-state offset.

Eliminating Derivative Action The effect of derivative action on the output may be eliminated by setting Td = 0 (most loops do not require a D parameter; it may make the loop unstable).

Eliminating Proportional Action Although rarely done, the effect of proportional term on the output may be eliminated by setting Kc = 0. Since Kc is also normally a multiplier of the integral coefficient (Ki) and the derivative coefficient (Kr), the CPU makes the computation of these values conditional on the value of Kc as follows:

$Ki = Kc * (Ts / Ti)$ if $Kc >< 0$

$Ki = Ts / Ti$ if $Kc = 0$ (I or ID only)

$Kr = Kc * (Td / Ts)$ if $Kc >< 0$

$Kr = Td / Ts$ if $Kc = 0$ (ID or D only)

Velocity Form of the PID Equation

The standard position form of the PID equation computes the actual actuator position. An alternative form of the PID equation computes the change in actuator position. This form of the equation is referred to as the velocity PID equation and is obtained by subtracting the equation at time "n" from the equation at time "n-1".

The velocity equation is given by:

$\Delta M_n = M - M_{n-1}$

$\Delta M_n = Kc * (e_n - e_{n-1}) + Ki * (PV_n - 2 * PV_{n-1} + PV_{n-2})$

Bumpless Transfer

The DL205 loop controller provides for bumpless mode changes. A bumpless transfer from manual mode to automatic mode is achieved by preventing the control output from changing immediately after the mode change.

When a loop is switched from Manual mode to Automatic mode, the setpoint and Bias are initialized as follows:

- **Position PID Algorithm**
  SP = PV
  Mx = M

  Velocity PID Algorithm
  SP = PV

The bumpless transfer feature of the DL205 is available in two types: Bumpless I and Bumpless II (see page 8-26). The transfer type is selected when the loop is set up.

Loop Alarms

The DL205 allows the user to specify alarm conditions that are to be monitored for each loop. Alarm conditions are reported to the CPU by setting up the alarms in ***Direct***SOFT using the PID setup alarm dialog when the loop is setup. The alarm features for each loop are:

- **PV Limit** – Specify up to four PV alarm points.

High-High	PV rises above the programmed High-High Alarm Limit.
High	PV rises above the programmed High Alarm Limit.
Low	PV fails below the Low Alarm Limit.
Low-Low	PV fails below the Low-Low Limit.

- **PV Deviation Alarm** – Specify an alarm for High and Low PV deviation from the setpoint (Yellow Deviation). An alarm for High-High and Low-Low PV deviation from the setpoint (Orange Deviation) may also be specified. When the PV is further from the setpoint than the programmed Yellow or Orange Deviation Limit, the corresponding alarm bit is activated.
- **Rate of Change** – This alarm is set when the PV changes faster than a specified rate-of-change limit.
- **PV Alarm Hysteresis** – The PV Limit Alarms and PV Deviation Alarms are programmed using threshold values. When the absolute value or deviation exceeds the threshold, the alarm status becomes true. Real-world PV signals have some noise on them, which can cause some fluctuation in the PV value in the CPU. As the PV value crosses an alarm threshold, its fluctuations will cause the alarm to be intermittent and annoy process operators. The solution is to use the PV Alarm Hysteresis feature.

Loop Operating Modes

The DL205 loop controller operates in one of three modes, either *Manual , Automatic* or *Cascade*.

Manual

In manual mode, the control output is determined by the operator, not the loop controller. While in manual mode, the loop controller will still monitor all of the alarms including High-High, High, Low, Low-Low, Yellow deviation, Orange deviation and Rate-of-Change.

Automatic

In automatic mode, the loop controller computes the control output based on the programmed parameters stored in V-memory. All alarms are monitored while in automatic.

Cascade

Cascade mode is an option with the DL205 PLC and is used in special control applications. If the cascade feature is used, the loop will operate as it would if in automatic mode except for the fact that a cascaded loop has a setpoint which is the control output from another loop.

Special Loop Calculations

Reverse Acting Loop

Although the PID algorithm is used in a direct, or forward, acting loop controller, there are times when a reverse acting control output is needed. The DL205 loop controller allows a loop to operate as reverse acting. With a reverse acting loop, the output is driven in the opposite direction of the error. For example, if SP > PV, then a reverse acting controller will decrease the output to increase the PV.

$$Mx = -Ki * e_n + Mx_{n-1}$$

$$M = -Kc * e_n + Kr(PV_n - PV_{n-1}) + Mx_n$$

Square Root of the Process Variable

Square root is selected whenever the PV is from a device such as an orifice meter which requires this calculation.

Error Squared Control

Whenever error squared control is selected, the error is calculated as:

$$e_n = (SP - PV_n) * ABS(SP - PV_n)$$

A loop using the error squared is less responsive than a loop using just the error, however, it will respond faster with a large error. The smaller the error, the less responsive the loop. Error squared control would typically be used in a PH control application.

Error Deadband Control

With error deadband control, no control action is taken if the PV is within the specified deadband area around the setpoint. The error deadband is the same above and below the setpoint.

Once the PV is outside of the error deadband around the setpoint, the entire error is used in the loop calculation.

$e_n = 0$ SP - Deadband Below_SP < PV < SP - Deadband_Above_SP
$e_n = P - PV_n$ otherwise

The error will be squared first if both Error Squared and Error Deadband are selected.

Derivative Gain Limiting

When the coefficient of the derivative term, Kr, is a large value, noise introduced into the PV can result in erratic loop output. This problem is corrected by specifying a derivative gain limiting coefficient, Kd. Derivative gain limiting is a first order filter applied to the derivative term computation, Y_n, as shown below.

$$Y_n = Y_{n-1} + \frac{Ts}{Ts + (\frac{Td}{Kd})} * (PV_n - Y_{n-1})$$

Position Algorithm

$$Mx = Ki * e_n + Mx_{n-1}$$
$$M = Kc * e_n - Kr * (Y_n - Y_{n-1}) + Mx$$

Velocity Algorithm

$$\Delta M = Kc * (e_n - e_{n-1}) + Ki * e_n - Kr * (Y_n - 2 * Y_{n-1} + Y_{n-2})$$

Ten Steps to Successful Process Control

Controllers such as the DL205 PLC provide sophisticated process control features. Automated control systems can be difficult to debug, because a given symptom can have many possible causes. We recommend a careful, step-by-step approach to bringing new control loops online:

Step 1: Know the Recipe

The most important is – how to produce your product. This knowledge is the foundation for designing an effective control system. A good process *recipe* will do the following:

- Identify all relevant Process Variables, such as temperature, pressure, or flow rates, etc., that need precise control.
- Plot the desired Setpoint values for each process variables for the duration of one process cycle.

Step 2: Plan Loop Control Strategy

This simply means choosing the method the machine will use to maintain control over the Process Variables to follow their Setpoints. This involves many issues and trade-offs, such as energy efficiency, equipment costs, ability to service the machine during production, and more. You must also determine how to generate the Setpoint value during the process, and whether a machine operator can change the SP.

Step 3: Size and Scale Loop Components

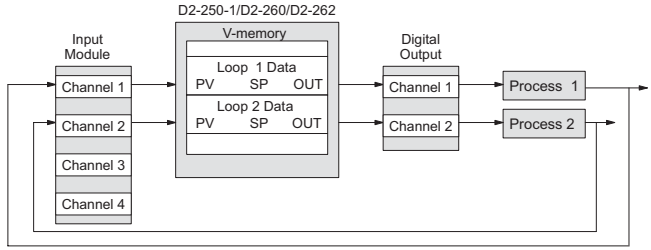
Assuming the control strategy is sound, it is still crucial to *properly size the actuator and properly scale the sensors.*

- Choose an actuator (heater, pump, etc) that matches the size of the load. An oversized actuator will have an overwhelming effect on your process after an SP change. However, an undersized actuator will allow the PV to lag or drift away from the SP after an SP change or process disturbance.
- Choose a PV sensor which matches the range of interest (and control) for our process. Decide the resolution of control you need for the PV (such as within 2°C), and make sure the sensor input value provides the loop with at least five times that resolution (at LSB level). However, an over-sensitive sensor can cause control oscillations, etc. The DL205 provides 12–bit and 15–bit unipolar and bipolar data format options, and a 16–bit unipolar option. This selection affects SP, PV, Control Output and Integrator sum.

Step 4: Select I/O Modules

After deciding the number of loops, PV variables to measure, and SP values, you can choose the appropriate I/O module. Refer to the figure on the next page. In many cases, you will be able to share input or output modules, or use an analog I/O combination module, among several control loops. The example shown sends the PV and Control Output signals for two loops through the same set of modules.

Automationdirect offers DL205 analog input modules with 4 channels per module that accept 0–20 mA or 4–20 mA signals. Also, analog input and output combination modules are now available. Thermocouple and RTD modules can also be used to maintain temperatures to a tenth of a degree. Refer to the sales catalog for further information on these modules, or find the modules on our website, **www.automationdirect.com.**



Step 5: Wiring and Installation

- After selection and procurement of all loop components and I/O module(s), you can perform the wiring and installation. Refer to the wiring guidelines in Chapter 2 of this manual, and to the **D2-ANLG-M** manual. The most common wiring errors when installing PID loop controls are:
- Reversing the polarity of sensor or actuator wiring connections, and
- Incorrect signal ground connections between loop components.

Step 6: Loop Parameters

After wiring and installation, choose the loop set-up parameters. The easiest method for programming the loop tables is using ***Direct***SOFT (5.0 or later). This software provides PID Setup using dialog boxes to simplify the task. **Note: It is important to understand the meaning of all loop parameters mentioned in this chapter before choosing values to enter.**

Step 7: Check Open Loop Performance

With the sensor and actuator wiring done, and loop parameters entered, we must manually and carefully check out the new control system using the Manual mode.

- Verify that the PV value from the sensor is correct.
- If it is safe to do so, gradually increase the control output up above 0%, and see if the PV responds (and moves in the correct direction!).

Step 8: Loop Tuning

If the Open Loop Test (page 8–40) shows the PV reading is correct and the control output has the proper effect on the process, you can follow the closed loop tuning procedure (see page 8–45). In this step, the loop is tuned so the PV automatically follows the SP.

Step 9: Run Process Cycle

If the closed loop test shows the PV will follow small changes in the SP, consider running an actual process cycle. You will need to have completed the programming which will generate the desired SP in real time. In this step, you may want to run a small test batch of product through the machine, watching the SP change according to the recipe.

WARNING: Be sure the Emergency Stop and power-down provision is readily accessible, in case the process goes out of control. Damage to equipment and/or serious injury to personnel can result from loss of control of some processes.

Step 10: Save Parameters

When the loop tests and tuning sessions are complete, be sure to save all loop set-up parameters to disk.

PID Loop Setup

Some Things to Do and Know Before Starting

Have your analog module installed and operational before beginning the loop setup (refer to the DL205 Analog Modules User Manual, D2-ANLG-M). The DL205 PLC gets its PID loop processing instructions from V-memory tables. There isn't a PID instruction that can be used in RLL, such as a block, to set up the PID loop control. Instead, the CPU reads the setup parameters from system V-memory locations. These locations are shown in the table below for reference only; they can be used in a RLL program if needed.

Address	Setup Parameter	Data type	Ranges	Read/Write
V7640	Loop Parameter Table Pointer	Octal	V1400 – V7340 V10000-V17740 (D2-250-1) V10000 - V37740 (D2-260/D2-262)	write
V7641	Number of Loops	BCD	1 – 4 (D2-250-1) 1 - 16 (D2-260/D2-262)	write
V7642	Loop Error Flags	BITS	0 or 1	read

NOTE: *The V-memory data is stored in SRAM memory. If power is removed from the CPU for an extended period of time, the PID Setup Parameters will be lost. It is recommended to use the optional battery backup to retain the memory in SRAM. Another option is to use the MOV instruction, which places the data in non-volatile memory, when setting up the parameters in the ladder program.*

PID Error Flags

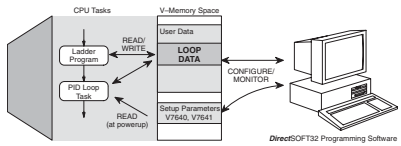
The CPU reports any programming errors of the setup parameters in V7640 and V7641. It does this by setting the appropriate bits in V7642 on program-to-run mode transitions.

If you use the *Direct*SOFT loop setup dialog box, its automatic range checking prohibits possible setup errors. However, the setup parameters may be written using other methods such as RLL, so the error flag register may be helpful in those cases. The following table lists the errors reported in V7642.

Bit	Error Description (0 = no error, 1 = error)
0	The starting address (in V7640) is out of the lower V-memory range.
1	The starting address (in V7640) is out of the upper V-memory range.
2	The number of loops selected (in V7641) is greater than 4 (D2-250-1) and 16 (D2-260/D2-262).
3	The loop table extends past (straddles) the boundary at V7377. Use an address closer to V1400.
4	Loop table extends past (straddles) the boundary at V17777 (D2-250-1) or V35777 (D2-260/D2-262). Use address closer to V10000.

As a quick check, if the CPU is in Run mode and V7642=0000, no programming errors.

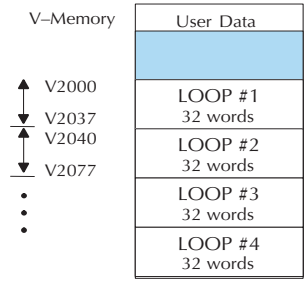
Establishing the Loop Table Size and Location



On a PROGRAM-to-RUN mode transition, the CPU reads the loop set-up parameters as pictured below. At that moment, the CPU learns the location of the loop table and the number of loops it configures. Then during the ladder program scan, the PID Loop task uses the loop data to perform calculations, generate alarms, and so on. There are some loop table parameters the CPU will read or write on every loop calculation.

***NOTE**: The DL205 CPU PID algorithm requires **Direct**SOFT and the D2-250-1, D2-260 or D2-262 CPUs with any firmware version. See our website for more information: **www.automationdirect.com**.*

The Loop Table contains data for only the number of loops that are selected. The address for the table is stored in V7641. Each loop configuration occupies 32 words (0 to 37 octal) in the loop table.

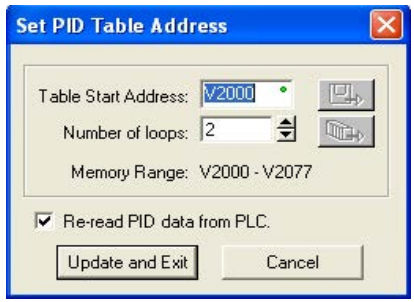


For example, consider an application with 4 loops, and V2000 has been chosen as the starting location. The Loop Parameter will occupy V2000 – V2037 for loop 1, V2040 – V2077 for loop 2 and so on. Loop 4 occupies V2140 - V2177.

Determine the block of V-memory to be used for each PID loop. Besides being the beginning of the PID parameter memory block, the first address will be the start of loop 1 parameters. Remember, there are 32 words (0 to 37 octal) needed for each loop. Once you have determined the beginning V-memory address to be used, you can set up and store the PID parameters either directly in your RLL program or by the using PID Setup in ***Direct***SOFT.

NOTE:** Whether one or more loops are being set up, this block of V-memory will only be used for the PID loop parameters. **Do not use this block of memory for anything else in your program.

Using ***Direct***SOFT is the simplest way to set-up the parameters. To set up the PID parameters, the DL205 must be powered up and connected to the programming computer. The parameters can only be entered in PID setup when the PLC is in the Program mode. Once the parameters have been entered and saved for each loop, changes made through the PID setup can be made, but only in Program Mode. You can type the beginning address in the PID Table Address dialog found when the PID Setup is opened in ***Direct***SOFT. This can be seen in the illustration below. After the address has been entered, the memory range will appear. Also, entering the number of PID loops (1 to 4 for the D2-250-1 and 16 for the D2-260 and D2-262) will set the total V-memory range for the number of loops entered. After the V-memory address has been entered, the necessary PID parameters for a basic loop operation for each loop can be setup with the dialogs made available.



***NOTE:** To access the Setup PID dialog, have an edited program open and click on PLC > Setup > PID..*

Loop Table Word Definitions

These are the loop parameters associated with each of the four loops available in the DL205. The parameters are listed in the following table. The address offset is in octal, to help you locate specific parameters in the loop table. For example, if a table begins at V2000, then the location of the reset (integral) term is Addr+11, or V2011. Do not use the Word # (in the first column) to calculate addresses.

Word #	Address+Offset	Description	Format	Read on-the-fly***
1	Addr + 0	PID Loop Mode Setting 1	Bits	Yes
2	Addr + 1	PID Loop Mode Setting 2		Yes
3	Addr + 2	Setpoint Value (SP)	Word/binary	Yes
4	Addr + 3	Process Variable (PV)		Yes
5	Addr + 4	Bias (Integrator) Value		Yes
6	Addr + 5	Control Output Value		Yes
7	Addr + 6	Loop Mode and Alarm Status	Bits	–
8	Addr + 7	Sample Rate Setting	Word/BCD	Yes
9	Addr + 10	Gain (Proportional) Setting		Yes
10	Addr + 11	Reset (Integral) Time Setting		Yes
11	Addr + 12	Rate (Derivative) Time Setting		Yes
12	Addr + 13	PV Value, Low-Low Alarm	Word/binary	No*
13	Addr + 14	PV Value, Low Alarm		No*
14	Addr + 15	PV Value, High Alarm		No*
15	Addr + 16	PV Value, High-High Alarm		No*
16	Addr + 17	PV Value, deviation alarm (YELLOW)		No*
17	Addr + 20	PV Value, deviation alarm (RED)		No*
18	Addr + 21	PV Value, rate-of-change alarm		No*
19	Addr + 22	PV Value, alarm hysteresis setting		No*
20	Addr + 23	PV Value, error deadband setting		Yes
21	Addr + 24	PV low-pass filter constant	Word/BCD	Yes
22	Addr + 25	Loop derivative gain limiting factor setting		No**
23	Addr + 26	SP value lower limit setting	Word/binary	Yes
24	Addr + 27	SP value upper limit setting		Yes
25	Addr + 30	Control output value lower limit setting		No**
26	Addr + 31	Control output value upper limit setting		No**
27	Addr + 32	Remote SP Value V-Memory Address Pointer	Word/hex	Yes
28	Addr + 33	Ramp/Soak Setting Flag	Bit	Yes
29	Addr + 34	Ramp/Soak Programming Table Starting Address	Word/hex	No**
30	Addr + 35	Ramp/Soak Programming Table Error Flags	Bits	No**
31	Addr + 36	PV auto transfer, channel number	Word/hex	Yes
32	Addr + 37	Control output auto transfer, channel number		Yes

* Read data only when alarm enable bit transitions from 0 to 1.
** Read data only on PLC Mode change.
*** Read on-the-fly means that the content of V-memory can be changed while the PID loop is in operation.

PID Mode Setting 1 Bit Descriptions (Addr + 00)

The individual bit definitions of the PID Mode Setting 1 word (Addr+00) are listed in the following table.

Bit	PID Mode Setting 1 Description	Read/Write	Bit=0	Bit=1
0	Manual Mode Loop Operation request	Write	–	01 request
1	Automatic Mode Loop Operation request		–	01 request
2	Cascade Mode Loop Operation request		–	01 request
3	Bumpless Transfer select		Mode I	Mode II
4	Direct or Reverse-Acting Loop select		Direct	Reverse
5	Position / Velocity Algorithm select		Position	Velocity
6	PV Linear / Square Root Extract select		Linear	Square root
7	Error Term Linear / Squared select		Linear	Squared
8	Error Deadband enable		Disable	Enable
9	Derivative Gain Limit select		Off	On
10	Bias (Integrator) Freeze select			
11	Ramp/Soak Operation select			
12	PV Alarm Monitor select			
13	PV Deviation alarm select			
14	PV rate-of-change alarm select			
15	Loop mode is independent from CPU mode when set		Loop with CPU mode	Loop Independent of CPU mode

PID Mode Setting 2 Bit Descriptions (Addr + 01)

The individual bit definitions of the PID Mode Setting 2 word (Addr+01) are listed in the following table.

Bit	PID Mode Setting 2 Word Description	Read/Write	Bit=0	Bit=1
0	Input (PV) and Control Output Range Unipolar/Bipolar select (See Notes 2 and 3)	Write	Unipolar	Bipolar
1	Input/Output Data Format select (See Notes 2 and 3)		12-bit	15-bit
2	Analog Input filter		Off	On
3	SP Input limit enable		Disable	Enable
4	Integral Gain (Reset) units select		Seconds	Minutes
5	Select Auto tune PID algorithm		Closed loop	Open loop
6	Auto tune selection		PID	PI only (rate = 0)
7	Auto tune start (See Note 1)	Read/write	Auto tune cancel/done	Force start
8	PID Scan Clock (internal use)	Read	–	–
9	Input/Output Data Format 16-bit select (See Notes 1 and 2)	Write	Not 16-bit	Select 16-bit
10	Select separate data format for input and output (See Notes 2, and 3)		Same format	Separate formats
11	Control Output Range Unipolar/Bipolar select See Notes 2, and 3)		Unipolar	Bipolar
12	Output Data Format select (See Notes 2, and 3)		12-bit	15-bit
13	Output data format 16-bit select (See Notes 2, and 3)		Not 16-bit	Select 16-bit
14–15	Reserved for future use	–	–	–

NOTE 1: *Bit 7 can be used to cancel Autotune mode by setting it to 0.*

NOTE 2: *If the value in bit 9 is 0, then the values in bits 0 and 1 are read. If the value in bit 9 is 1, then the values in bits 0 and 1 are not read, and bit 9 defines the data format (the range is automatically unipolar).*

NOTE 3: *If the value in bit 10 is 0, then the values in bits 0, 1 and 9 define the input and output ranges and data formats (the values in bits 11, 12, and 13 are not read). If the value in bit 10 is 1, then the values in bits 0, 1, and 9 define only the input range and data format, and bits 11, 12, and 13 are read and define the output range and data format.*

NOTE 4: *If bit 10 has a value of 1 and bit 13 has a value of 0, then bits 11 and 12 are read and define the output range and data format. If bit 10 and bit 13 each have a value of 1, then bits 11 and 12 are not read, and bit 13 defines the data format, (the output range is automatically unipolar).*

Mode/Alarm Monitoring Word (Addr + 06)

The individual bit definitions of the Mode / Alarm monitoring (Addr+06) word are listed in the following table.

Bit	Mode/Alarm Bit Description	Read/Write	Bit=0	Bit=1
0	Manual Mode Indication	Read	–	Manual
1	Automatic Mode Indication		–	Auto
2	Cascade Mode Indication		–	Cascade
3	PV Input Low–Low Alarm		Off	On
4	PV Input Low Alarm			
5	PV Input High Alarm			
6	PV Input High–High Alarm			
7	PV Input YELLOW Deviation Alarm			
8	PV Input RED Deviation Alarm			
9	PV Input Rate-of-Change Alarm			
10	Alarm Value Programming Error		–	Error
11	Loop Calculation Overflow/Underflow		–	Error
12	Loop in Auto-Tune indication		Off	On
13	Auto-Tune error indication		–	Error
14–15	Reserved for Future Use	–	–	–

Ramp/Soak Table Flags (Addr + 33)

The individual bit definitions of the Ramp/Soak Table Flag (Addr+33) word are listed in the following table.

Bit	Ramp/Soak Flag Bit Description	Read/Write	Bit=0	Bit=1
0	Start Ramp / Soak Profile	Write	–	01 Start
1	Hold Ramp / Soak Profile		–	01 Hold
2	Resume Ramp / soak Profile		–	01 Resume
3	Jog Ramp / Soak Profile		–	01 Jog
4	Ramp / Soak Profile Complete	Read	–	Complete
5	PV Input Ramp / Soak Deviation		Off	On
6	Ramp / Soak Profile in Hold		Off	On
7	Reserved		–	–
8–15	Current Step in R/S Profile		Decode as byte (hex)	

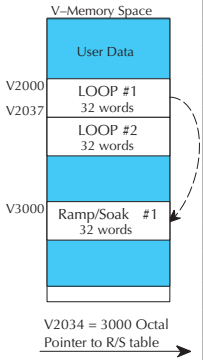
Bits 8–15 must be read as a byte to indicate the current segment number of the Ramp/Soak generator in the profile. This byte will have the values 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, and 10, which represent segments 1 to 16 respectively. If the byte=0, then the Ramp/Soak table is not active.

Ramp/Soak Table Location (Addr + 34)

Each loop that you configure has the option of using a built-in Ramp/Soak generator dedicated to that loop. This feature generates SP values that follow a profile. To use the Ramp Soak feature, you must program a separate table of 32 words with appropriate values. The *Direct*SOFT dialog box makes this easy to do.

In the loop table, the Ramp/Soak Table Pointer at Addr+34 must point to the start of the ramp/soak data for that loop. This may be anywhere in user memory, and does not have to adjoin to the Loop Parameter table, as shown to the left. Each R/S table requires 32 words, regardless of the number of segments programmed.

The ramp/soak table parameters are defined in the table below. Further details are in the section on Ramp/Soak Operation in this chapter.



Addr Offset	Step	Description	Addr Offset	Step	Description
+ 00	1	Ramp End SP Value	+ 20	9	Ramp End SP Value
+ 01	1	Ramp Slope	+ 21	9	Ramp Slope
+ 02	2	Soak Duration	+ 22	10	Soak Duration
+ 03	2	Soak PV Deviation	+ 23	10	Soak PV Deviation
+ 04	3	Ramp End SP Value	+ 24	11	Ramp End SP Value
+ 05	3	Ramp Slope	+ 25	11	Ramp Slope
+ 06	4	Soak Duration	+ 26	12	Soak Duration
+ 07	4	Soak PV Deviation	+ 27	12	Soak PV Deviation
+ 10	5	Ramp End SP Value	+ 30	13	Ramp End SP Value
+ 11	5	Ramp Slope	+ 31	13	Ramp Slope
+ 12	6	Soak Duration	+ 32	14	Soak Duration
+ 13	6	Soak PV Deviation	+ 33	14	Soak PV Deviation
+ 14	7	Ramp End SP Value	+ 34	15	Ramp End SP Value
+ 15	7	Ramp Slope	+ 35	15	Ramp Slope
+ 16	8	Soak Duration	+ 36	16	Soak Duration
+ 17	8	Soak PV Deviation	+ 37	16	Soak PV Deviation

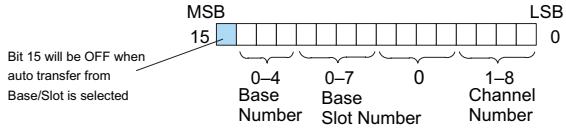
Ramp/Soak Table Programming Error Flags (Addr + 35)

The individual bit definitions of the Ramp/Soak Table **Programming Error Flags** word (Addr+35) are listed in the following table. Further details are given in the PID Loop Mode section and in the PV Alarm section later in chapter 8.

Bit	R/S Error Flag Bit Description	Read/Write	Bit=0	Bit=1
0	Starting Addr out of lower V-memory range	Read	–	Error
1	Starting Addr out of upper V-memory range	Read	–	Error
2–3	Reserved for Future Use	–	–	–
4	Starting Addr in System Parameter V-memory Range	Read	–	Error
5–15	Reserved for Future Use	–	–	–

PV Auto Transfer (Addr + 36) from I/O Module Base/Slot/Channel Option

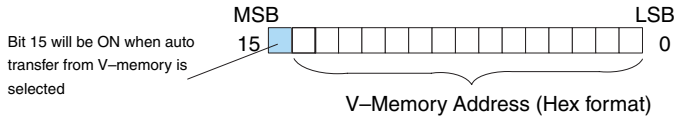
The nibble definitions for PV Auto Transfer word (Addr + 36) are listed in the table below for the Transfer from Base/Slot option. **When this option is used for any channel on an analog input module, the ladder logic pointer method cannot be used for this module.** Refer to the DL205 Analog I/O Modules (D2-ANLG-M) for pointer method information.



CPU	Base Number	Base Slot Number	Channel Number
D2-250-1	Local CPU base = 0 Local expansion base = 1-2	0-7	1-8
D2-260/D2-262	Local CPU base = 0 Local expansion base = 1-4	0-7	1-8

PV Auto Transfer (Addr + 36) from V-memory Option

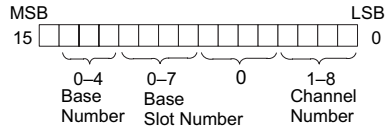
The nibble definitions for PV Auto Transfer word (Addr + 36) are listed in the table below for the Transfer from V-memory option. The ladder logic pointer method can be used with this option to get the analog module's channel values into V-memory. Refer to the DL205 Analog I/O Modules (D2-ANLG-M) for pointer method information.



Memory Type	D2-250-1 Range	D2-260/D2-262 Range
V-memory	V1400-V7377 V10000-1777	V400-V677 V1400-V7377 V10000-V35777

Control Output Auto Transfer (Addr + 37)

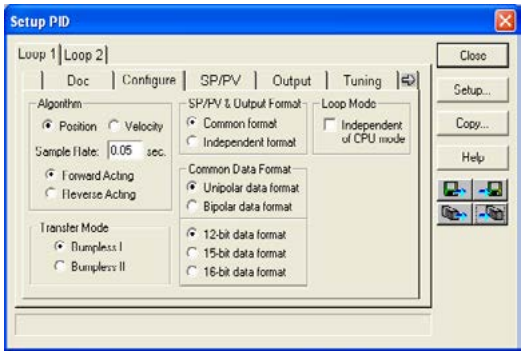
The nibble definitions for Control Output Auto Transfer word (Addr + 37) are listed in the table below. **When the Control Output Auto Transfer function is used for any channel on an analog input module, the ladder logic pointer method cannot be used for this module.** Refer to the DL205 Analog I/O Modules (D2-ANLG-M) for pointer method information.



CPU	Base Number	Base Slot Number	Channel Number
D2-250-1	Local CPU base = 0 Local expansion base = 1-2	0-7	1-8
D2-260/D2-262	Local CPU base = 0 Local expansion base = 1-4	0-7	1-8

Configure the PID Loop

Once the PID table is established in V-memory, configuring the PID loop continues with the ***Direct***SOFT PID set-up configuration dialog. You will need to check and fill in the data required to control the PID loop. Select Configure and the following dialog box will appear for this process.

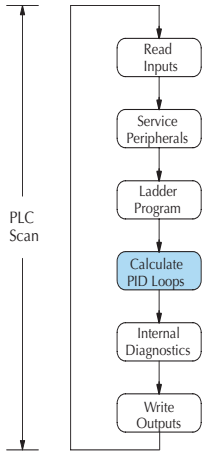


Select the Algorithm Type

Chose either *Position* or *Velocity*. The default algorithm is Position. This is the choice for most applications which include heating and cooling loops as well as most position and level control loops. A typical velocity control will consist of a process variable such as a flow totalizer in a flow control loop.

Enter the Sample Rate

The main tasks of the CPU fall into categories as shown to the right. The list represents the tasks done when the CPU is in Run Mode, on each PLC scan. Note that PID loop calculations occur after the ladder logic task.



The *sample rate* of a control loop is simply the frequency of the PID calculation. Each calculation generates a new control output value. With the DL205 CPUs, you can set the sample rate of a loop from 50 ms to 99.99 seconds. Most loops do not require a fresh PID calculation on every PLC scan. Some loops may need to be calculated only once in 1000 scans.

Enter 0.05 sec., or the sample rate of your choice, for each loop, and the CPU automatically schedules and executes PID calculations on the appropriate scans.

NOTE: *If more than 4 loops are programmed, enter a minimum of 0.1 second.*

Select Forward/Reverse

It is important to know in which direction the control output will respond to the error (SP-PV), either *forward* or *reverse.* A forward (direct) acting control loop means that whenever the control output increases, the process variable will also increase. The control outputs of most PID loops are forward acting, such as a heating control loop. An increase in heat applied will increase the PV (temperature).

A reverse acting control loop is one where an increase in the control output results in a decrease in the PV. A common example of this would be a refrigeration system, where an increase in the cooling input causes a decrease in the PV (temperature).

The Transfer Mode

Choose either Bumpless I or Bumpless II to provide a smooth transition of the control output from Manual Mode to Auto Mode. Choosing Bumpless I will set the SP equal to the PV when the control output is switched from Manual to Auto. If this is not desired, choose BumplessII.

The characteristics of Bumpless I and II transfer types are listed in the chart below. Note that their operation also depends on which PID algorithm you are using, the position or velocity form of the PID equation. Note that you must use Bumpless Transfer type I when using the velocity form of the PID algorithm.

Transfer Type	Transfer Select Bit 3	PID Algorithm	Manual-to-Auto Transfer Action	Auto-to-Cascade Transfer Action
Bumpless Transfer I	0	Position	Forces Bias = Control Output Forces SP = PV	Forces Major Loop Output = Minor Loop PV
		Velocity	Forces SP = PV	Forces Major Loop Output = Minor Loop PV
Bumpless Transfer II	1	Position	Forces Bias = Control Output	None
		Velocity	None	None

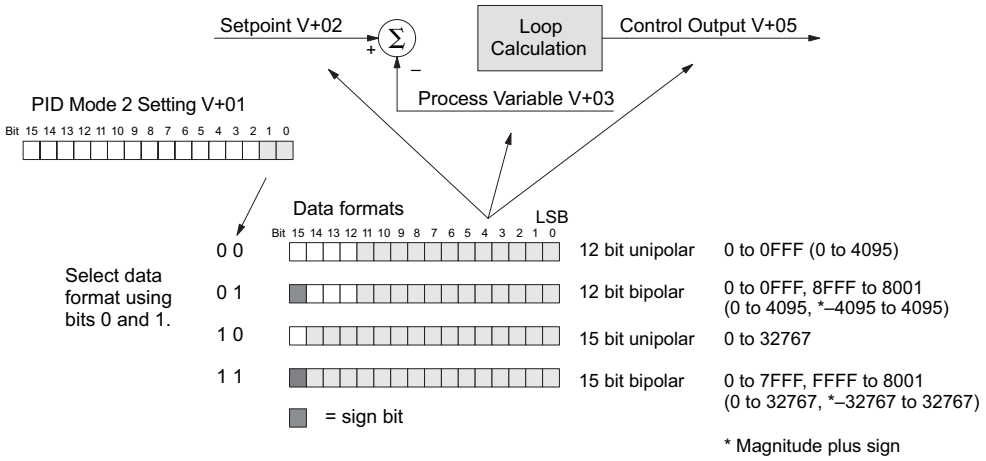
The transfer type can also be selected in an RLL program by setting bit 3 of PID Mode Setting 1, V+00 setting as shown.

SP/PV & Output Format

This block allows you to select either *Common format* or *Independent format*. Common format is the default and is most commonly used. With this format, both SP/PV and Output will have the same data structure. Both will have the same number of bits and either bipolar or unipolar. If Independent format is selected, the data structure selections will be grayed out. The reason for this is that they become independently selectable in the *SP/PV* and the *Output* dialogs.

Common Data Format

Select either *Unipolar data format* (which is positive data only) in 12 bit (0 to 4095), 15 bit (0 to 32767), or 16 bit (0 to 65535) format, or *Bipolar data format,* which ranges from negative to positive (-4095 to 4095 or -32767 to 32767) and requires a sign bit. Bipolar selection displays input/output as magnitude plus sign, not two's complement. The bipolar selection is not available when 16-bit data format is selected.



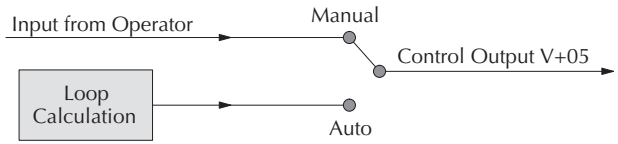
The data format determines the numerical interface between the PID loop and the PV sensor and the control output device. This selects the data format for both the SP and the PV.

Loop Mode

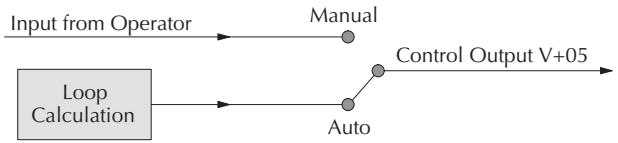
Loop Mode is a special feature that allows the PID loop controller to perform closed-loop control while the CPU is in the Program Mode. Careful thought must be taken before using this feature called *Independent of CPU mode* in the dialog. Before continuing with the PID setup, a knowledge of the three PID loop modes will be helpful.

The DL205 provides the three standard control modes: *Manual, Automatic,* and *Cascade*. The sources of the three basic variables, SP, PV and control output, are different for each mode.

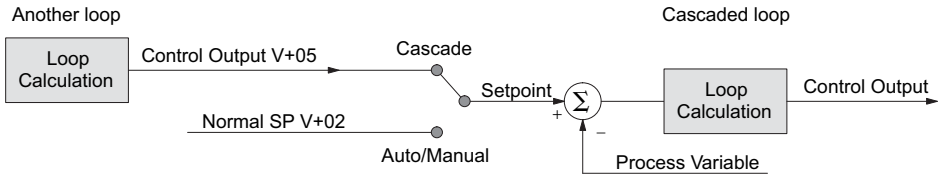
In Manual Mode, the loop is not executing PID calculations (however, loop alarms are still active). With regard to the loop table, the CPU stops writing values to location V+05 (control output) for that loop. *It is expected that an operator or other intelligent source* is manually controlling the output by observing the PV and writing data to the control output as necessary to keep the process under control. The drawing below shows the equivalent schematic diagram of manual mode operation.



In Automatic Mode, the loop operates normally and generates new control output values. It calculates the PID equation and writes the result in location V+05 every sample period of that loop. The equivalent schematic diagram is shown below.

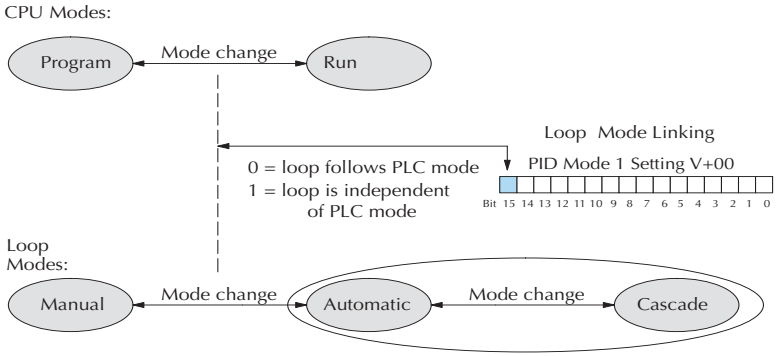


In Cascade Mode, the loop operates as it does in Automatic Mode, with one important difference. The data source for the SP changes from its normal location at V+02 to using the control output value, V+05, from another loop. So in Auto or Manual modes, the loop calculation uses the data at V+02. In Cascade Mode, the loop calculation reads the control output from another loop's parameter table, V+05.



As pictured below, a loop can be changed from one mode to another, but *cannot go from Manual Mode directly to Cascade, or vice versa*. This mode change is prohibited because a loop would be changing two data sources at the same time and could cause a loss of control.

Once the CPU is operating in the Run Mode, the normal operation of the PID loop controller is to read the loop data and perform calculations on each scan of the RLL program. When the CPU is placed in the Program Mode, the RLL program halts operation and all PID loops are automatically put into the Manual Mode. The PID parameters can then be changed if desired. Similarly, by placing the CPU in the Run mode, the PID loops are returned to the operational mode which they were previously in, i.e., Manual, Automatic and Cascade. With this selection, you automatically affect the modes by changing the CPU mode.



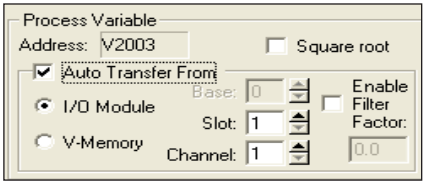
If bit 15 is set to one, then the loops will run independently of the CPU mode. It is like having two independent processors in the CPU... one is running the RLL program and the other is running the process loops.

Having the ability to run loops independently of the RLL program makes it feasible to make a ladder logic change while the process is still running. This is especially beneficial for large-mass continuous processes that are difficult or costly to interrupt. The independent of CPU is the feature used for this.

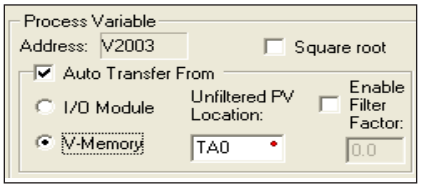
If you need to operate the PID loops while the RLL program is halted, in Program Mode, either select the Independent of CPU mode in the dialog or edit your program to set and reset bit 15 of PID Mode Setting 1 word (V+00) in your RLL program. If the bit is set to a zero, the loop will follow the CPU mode; then, when the CPU is placed in the Program Mode, all loops will be forced into the Manual Mode.

When Independent of CPU mode is used, you should also set the PV to be read directly from an analog input module. This can easily be done in the PID set-up dialog, SP/PV.

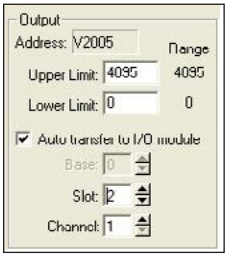
The SP/PV dialog has a block entitled *Process Variable*. There is a block within this block called *Auto Transfer From* (from analog input) with the information grayed out. Checking the box to the left of the Auto Transfer From will highlight the information. Select *I/O Module* then enter the slot number in which the input module resides. Next, select the analog input channel of your choice.



The second choice is *V-Memory.* When this is selected, the V-memory address from where the PV is transferred must be specified.



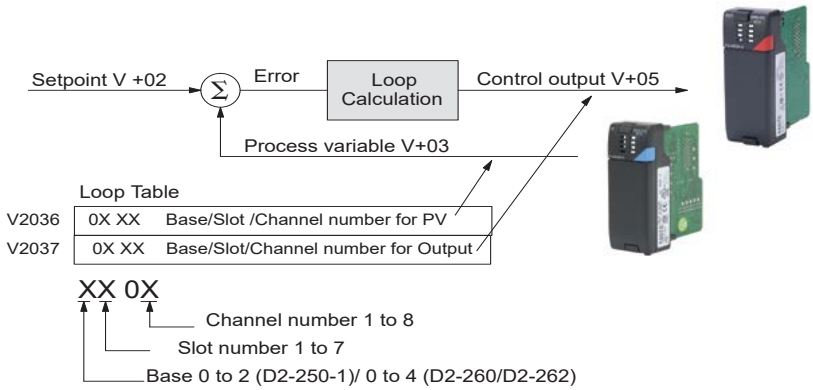
Whichever method of auto transfer is used, it is recommended to check the *Enable Filter Factor* (a low pass filter) and specify the coefficient.



You should also select the analog output for the control output to be transferred to. This is done in the PID setup *Output* dialog shown here. The block of information in this dialog is grayed-out until the box next to *Auto transfer to I/O module* is checked. Once checked, enter the slot number where the output module is residing and then enter the analog output channel number.

NOTE: *To make changes to any loop table parameters, the PID loop must be in Manual mode and the PLC must be stopped. If you have selected to operate the PID loop independent of the CPU mode, then you must take certain steps to make it possible to make loop parameter changes. You can temporarily make the loops follow the CPU mode by changing bit 15 to 0. Then, you will be able to place the loop into Manual Mode using* ***Direct****SOFT. After you change the loop's parameter settings, restore bit 15 to a value of 1 to re-establish PID operation independent of CPU.*

You may optionally configure each loop to access its analog I/O (PV and control output) by placing proper values in the associated loop table registers in your RLL program. The following figure shows the loop table parameters at V+36 and V+37 and their auto transfer role to access the analog values directly

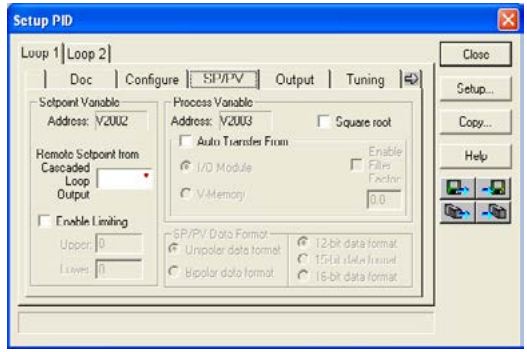


When these loop table parameters are programmed directly, a value of **0102** in register V2036 directs the loop controller to read the PV data from channel 2 of the analog input module in slot 1. A value of **0000** in either register tells the loop controller not to access the corresponding analog value directly. In that case, ladder logic must be used to transfer the value between the analog input and the loop table.

NOTE: *When auto transfer to/from I/O is used, the analog data for all of the channels on the analog module cannot be accessed by any other method, i.e., pointer or multiplex.*

SP/PV Addresses

An SP/PV dialog will be made available to set up how the setpoint (SP) and the process variable (PV) will be used in the loop. If this loop is the minor loop of a cascaded pair, enter that control output address in the *Remote SP from Cascaded Loop Output* area. It is sometimes desirable to limit the range of setpoint values allowed to be entered. To activate this feature, check the box next to *Enable Limiting*. This will activate the *Upper* and *Lower* fields for the values to be entered.

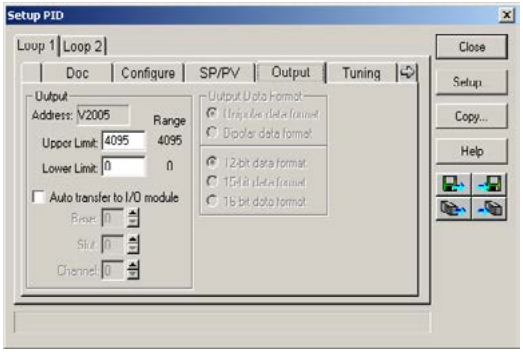


Set the limits around the SP value to prevent an operator from entering a setpoint value outside of a safe range. The *Square root* box is only checked for certain PID loops, such as a flow control loop. If the *Auto transfer from I/O module* is selected, a first-order low-pass filter can be used by checking the *Enable Filter* box. The filter coefficient is user specified. The use of this filter is recommended during closed loop auto-tuning. If the Independent format had been checked previously, make the Data format selections here.

NOTE: *The SP/PV dialog can be left as it first appears for basic PID operation.*

Set Control Output Limits

Another dialog that will be available in the PID setup will be the Output dialog. The control output address, V+05, (determined by the PID loop table beginning address) will be in view. Enter the output range limits, *Upper Limit* and *Lower Limit,* that will meet the requirement of the process and which will agree with the data format that has been selected. For a basic PID operation using a 12-bit output module, set the Upper Limit to 4095 and leave the Lower Limit set to 0.



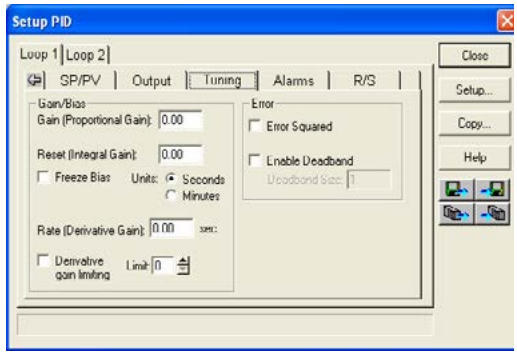
Check the box next for *Auto transfer to I/O module* if there is a need to send the control output to a certain analog output module, as in the case of using the Loop Mode independent of CPU Mode; otherwise, the PID output signal cannot control the analog output when the PLC is not in RUN Mode. If the *Auto transfer to I/O module* feature is checked, all channels of the module must be used for PID control outputs. If Independent format has been previously chosen, the *Output Data Format* will need to be setup here, that is, select Unipolar or Bipolar format and the bit structure. This area is not available and is grayed out if *Common format* has been chosen (see page 8-26).

WARNING: If the Upper Limit is set to zero, the output will never get above zero. In effect, there will be no control output.

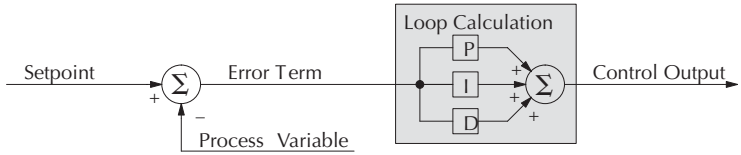
NOTE: *When auto transfer to/from I/O is used, the analog data for all of the channels on the analog module cannot be accessed by any other method, i.e., pointer or multiplex.*

Enter PID Parameters

Another PID setup dialog, Tuning, is for entering the PID parameters shown as: Gain (Proportional Gain), Reset (Integral Gain) and Rate (Derivative Gain).



Recall the position and velocity forms of the PID loop equations which were introduced earlier. The equations basically show the three components of the PID calculation: Proportional Gain (P), Integral Gain (I) and Derivative Gain (D). The following diagram shows a form of the PID calculation in which the control output is the sum of the proportional gain, integral gain and derivative gain. With each calculation of the loop, each term receives the same error signal value.



The P, I and D gains are 4-digit BCD numbers with values from 0000 to 9999. They contain an implied decimal point in the middle, so the values are actually 00.00 to 99.99. Some gain values have units – Proportional gain has no unit, Integral gain may be selected in seconds or in minutes, and Derivative gain is in seconds.

Gain (Proportional Gain) – This is the most basic gain of the three. Values range from 0000 to 9999, but they are used internally as xx.xx. An entry of "0000" effectively removes the proportional term from the PID equation. This accommodates applications which need integral-only loops.

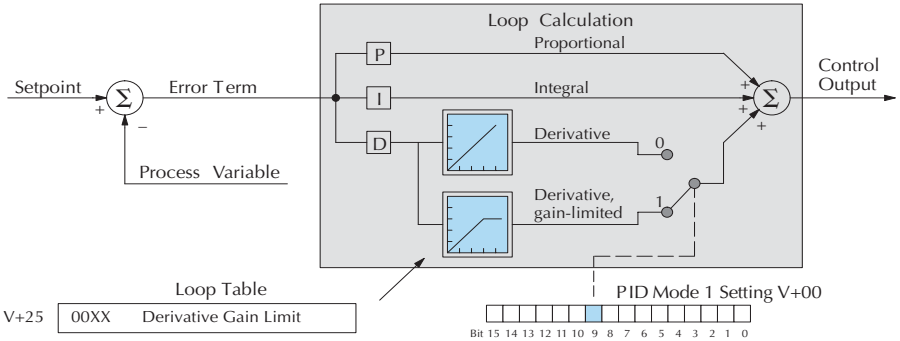
Reset (Integral Gain) – Values range from 0001 to 9998, but they are used internally as xx.xx. An entry of "0000" or "9999" causes the integral gain to be "infinity", effectively removing the integrator term from the PID equation. This accommodates applications that need proportional-only loops. The units of integral gain may be either seconds or minutes, as shown in the above dialog.

Rate (Derivative Gain) – Values which can be entered range from 0001 to 9999, but they are used internally as XX.XX. An entry of "0000" allows removal of the derivative term from the PID equation (a common practice). This accommodates applications that require only proportional and/or integral loops. Most control loops will operate as a PI loop.

NOTE: *You may elect to leave the tuning dialog blank and enter the tuning parameters in the DirectSOFT PID View.*

Derivative Gain Limiting

The derivative gain (rate) has an optional gain-limiting feature. This is provided because the derivative gain reacts badly to PV signal noise or other causes of sudden PV fluctuations. The function of the gain-limiting is shown in the diagram below.



The gain limit can be particularly useful during loop tuning. Most loops can tolerate only a little derivative gain without going into uncontrolled oscillations.

If this option is checked, a *Limit* from 0 to 20 must also be entered.

NOTE: *When first configuring a loop, it's best to use the standard error term until after the loop is tuned. Once the loop is tuned, you will be able to tell if these functions will enhance control. The Error Squared and/or Enable Deadband can be selected later in the PID setup. Also, values are not required to be entered in the Tuning dialog, but they can set later in the* ***Direct****SOFT PID View.*

Error Term Selection

The error term is internal to the CPUs PID loop controller, and is generated again in each PID calculation. Although its data is not directly accessible, you can easily calculate it by subtracting: Error = (SP–PV). If the PV square-root extract is enabled, then: Error = =PV In any case, the size of the error and algebraic sign determine the next change of the control output for each PID calculation.

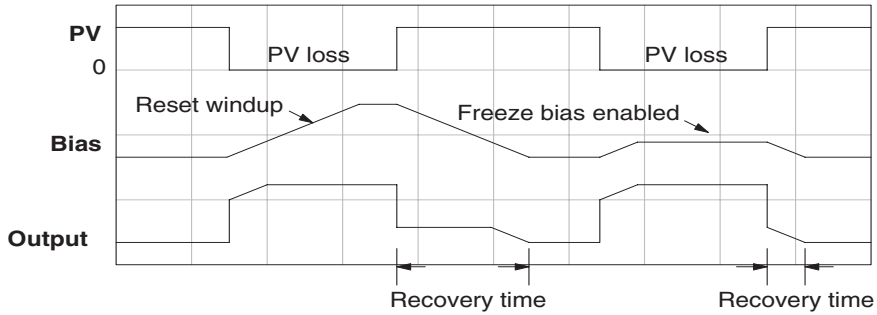
Error Squared – When selected, the squared error function simply squares the error term (but preserves the original algebraic sign), which is used in the calculation. This affects the Control Output by diminishing its response to smaller error values, but maintaining its response to larger errors. Some situations in which the error squared term might be useful:

- Noisy PV signal – using a squared error term can reduce the effect of low-frequency electrical noise on the PV, which will make the control system jittery. A squared error maintains the response to larger errors.
- Non-linear process – some processes (such as chemical pH control) require non-linear controllers for best results. Another application is surge tank control, where the Control Output signal must be smooth.

Enable Deadband – When selected, the enable deadband function takes a range of small error values near zero, and simply substitutes zero as the value of the error. If the error is larger than the deadband range, then the error value is used normally.

Freeze Bias

The term reset windup refers to an undesirable characteristic of integrator behavior which occurs naturally under certain conditions. Refer to the figure below. Suppose the PV signal becomes disconnected, and the PV value goes to zero. While this is a serious loop fault, it is made worse by reset windup. Notice the bias (reset) term keeps integrating normally during the PV disconnect, until its upper limit is reached. When the PV signal returns, the bias value is saturated (windup) and takes a long time to return to normal. The loop output consequently has an extended recovery time. Until recovery, the output level is wrong and causes further problems.



In the second PV signal loss episode in the figure, the freeze bias feature is enabled. It causes the bias value to freeze when the control output goes out of bounds. Much of the reset windup is thus avoided, and the output recovery time is much less.

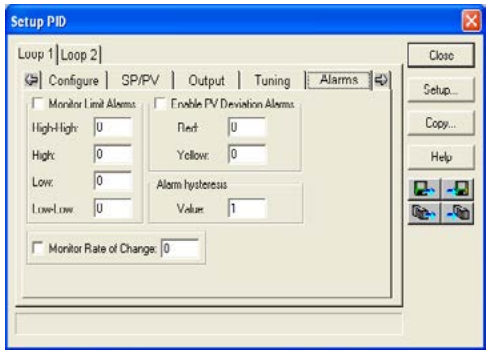
For most applications, the freeze bias feature will work with the loop as described above. It is suggested to enable this feature by selecting it in the dialog. Bit 10 of PID Mode Setting 1 (V+00) word can also be set in RLL.

NOTE: *The freeze bias feature stops the bias term from changing when the control output reaches the end of the data range. If you have set limits on the control output other than the range (i.e., 0–4095 for a unipolar/12-bit loop), the bias term still uses the end of range for the stopping point and bias freeze will not work.*

Set Up the PID Alarms

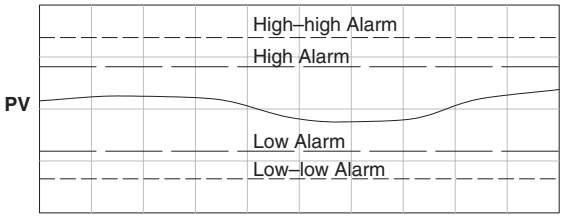
Although the setup of the PID alarms is optional, you surely would not want to operate a process without monitoring it. The performance of a process control loop may generally be measured by how closely the process variable matches the setpoint. Most process control loops in industry operate continuously, and will eventually lose control of the PV due to an error condition. Process alarms are vital in the early discovery of a loop error condition and can alert plant personnel to manually control a loop or take other measures until the error condition has been repaired.

The alarm thresholds are fully programmable, and each type of alarm may be independently enabled and monitored. The following diagram shows the Alarm dialog in the PID setup, which simplifies the alarm setup.



Monitor Limit Alarms

Checking this box will allow all of the PV limit alarms to be monitored once the limits are entered. The PV absolute value alarms are organized as two upper and two lower alarms. The alarm status is false as long as the PV value remains in the region between the upper and lower alarms, as shown below. The alarms nearest the safe zone are named *High Alarm* and *Low Alarm.* If the loop loses control, the PV will cross one of these thresholds first. Therefore, you can program the appropriate alarm threshold values in the loop table locations shown below to the right. The data format is the same as the PV and SP (12-bit or 15-bit). The threshold values for these alarms should be set to give an operator an early warning if the process loses control.



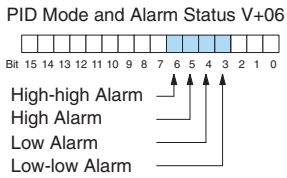
Loop Table

V+16	XXXX	High-high Alarm
V+15	XXXX	High Alarm
V+14	XXXX	Low Alarm
V+13	XXXX	Low-low Alarm

NOTE: *The Alarm dialog can be left as it first appears, without alarm entries. The alarms can then be setup in the* ***Direct****SOFT PID View.*

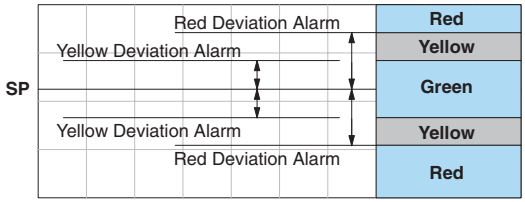
If the process remains out of control for some time, the PV will eventually cross one of the outer alarm thresholds, named High-high alarm and Low-low alarm. Their threshold values are programmed using the loop table registers listed above. A High-high or Low-low alarm indicates a serious condition exists, and needs the immediate attention of the operator.

The PV Absolute Value Alarms are reported in the four bits in the PID Mode and Alarm Status word in the loop table, as shown to the right. We highly recommend using ladder logic to monitor these bits. The bit-of-word instructions make this easy to do. Additionally, you can monitor PID alarms using *Direct*SOFT.



PV Deviation Alarms

The PV Deviation Alarms monitor the PV deviation with respect to the SP value. The deviation alarm has two programmable thresholds, and each threshold is applied equally above and below the current SP value. In the figure below, the smaller deviation alarm is called the **Yellow Deviation**, indicating a cautionary condition for the loop. The larger deviation alarm is called the **Red Deviation**, indicating a strong error condition for the loop. The threshold values use the loop parameter table locations V+17 and V+20 as shown.

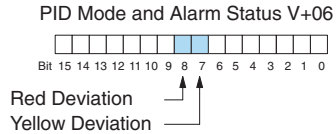


Loop Table

V+17	XXXX	Yellow Deviation Alarm
V+20	XXXX	Red Deviation Alarm

The thresholds define zones, which fluctuate with the SP value. The green zone which surrounds the SP value represents a safe (no alarm) condition. The yellow zones lie outside the green zone, and the red zones are beyond those.

The PV Deviation Alarms are reported in the two bits in the PID Mode and Alarm Status word in the loop table, as shown to the right. We highly recommend using ladder logic to monitor these bits. The bit-of-word instructions make this easy to do. Additionally, you can monitor PID alarms using *Direct*SOFT.



The PV Deviation Alarm can be independently enabled and disabled from the other PV alarms, using bit 13 of the PID Mode Setting 1 V+00 word.

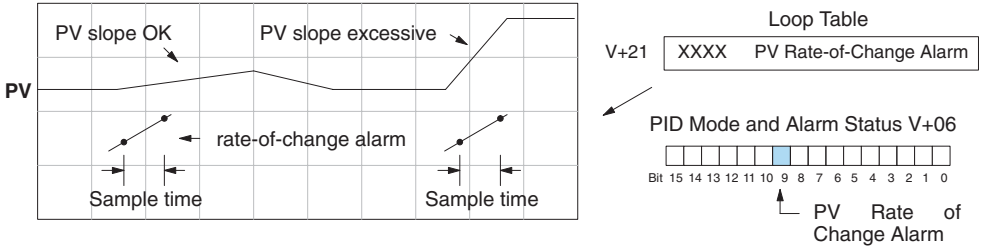
Remember the alarm hysteresis feature works in conjunction with both the deviation and absolute value alarms, and is discussed at the end of this section.

NOTE: *PID deviation alarm only works in Auto mode.*

PV Rate-of-Change Alarm

An excellent way to get an early warning of a process fault is to monitor the *rate-of-change* of the PV. Most batch processes have large masses and slowly-changing PV values. A relatively fast-changing PV will result from a broken signal wire for either the PV or control output, an SP value error, or other causes. If the operator responds to a PV Rate-of-Change Alarm quickly and effectively, the PV absolute value will not reach the point where the material in process would be ruined.

The DL205 loop controller provides a programmable PV Rate-of-Change Alarm, as shown below. The rate-of-change is specified in PV units change per loop sample time. This value is programmed into the loop table location V+21.



As an example, suppose the PV is the temperature for your process, and you want an alarm whenever the temperature changes faster than 15 degrees/minute. The PV counts per degree and the loop sample rate must be known. Then, suppose the PV value (in V+03 location) represents 10 counts per degree, and the loop sample rate is 2 seconds. Use the formula below to convert our engineering units to counts/sample period:

$$\text{Alarm Rate-of-Change} = \frac{15 \text{ degrees}}{1 \text{ minute}} \times \frac{10 \text{ counts / degree}}{30 \text{ loop samples / min.}}$$

$$= \frac{150}{30} = 5 \text{ counts / sample period}$$

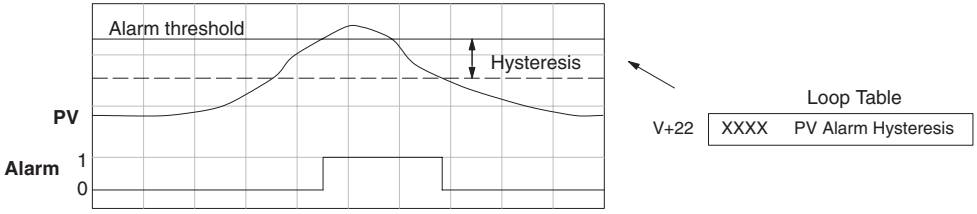
From the calculation result, you would program the value 5 in the loop table for the rate-of-change. The PV Rate-of-Change Alarm can be independently enabled and disabled from the other PV alarms, using bit 14 of the PID Mode Setting 1 V+00 word.

The alarm hysteresis feature (discussed next) does not affect the Rate-of-Change Alarm.

PV Alarm Hysteresis

The PV Absolute Value Alarm and PV Deviation Alarm are programmed using threshold values. When the absolute value or deviation exceeds the threshold, the alarm status becomes true. Real-world PV signals have some noise on them, which can cause some fluctuation in the PV value in the CPU. As the PV value crosses an alarm threshold, its fluctuations cause the alarm to be intermittent and annoy process operators. The solution is to use the PV Alarm Hysteresis feature.

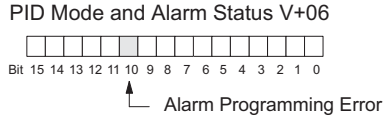
The PV Alarm Hysteresis amount is programmable from 1 to 200 (binary/decimal). When using the PV Deviation Alarm, the programmed hysteresis amount must be less than the programmed deviation amount. The figure below shows how the hysteresis is applied when the PV value goes past a threshold and descends back through it.



The hysteresis amount is applied after the threshold is crossed, and toward the safe zone. In this way, the alarm activates immediately above the programmed threshold value. It delays turning off until the PV value has returned through the threshold by the hysteresis amount.

Alarm Programming Error

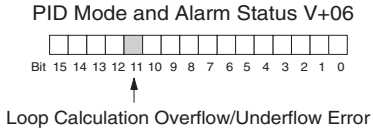
The PV Alarm threshold values must have certain mathematical relationships to be valid. The requirements are listed below. If not met, the Alarm Programming Error bit will be set, as indicated to the right.



- PV Absolute Alarm value requirements:
  Low-low < Low < High < High-high
- PV Deviation Alarm requirements:
  Yellow < Red

Loop Calculation Overflow/Underflow Error

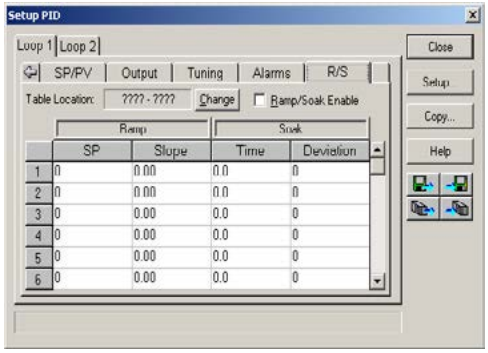
This error occurs whenever the output reaches its upper or lower limit and the PV does not reach the setpoint. A typical example might be when a valve is stuck, the output is at its limit, but the PV has not reached setpoint.



NOTE: *Overflow/underflow can be alarmed in PID View. The optional C-more operator interface panel (see the automationdirect.com website) can also be set up to read these error bits using the PID Faceplate templates.*

Ramp/Soak

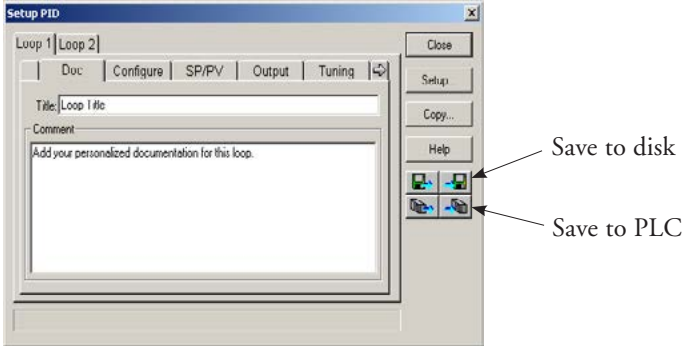
R/S (Ramp/Soak) is the last dialog available in the PID setup. The basic PID does not require any entries to be made in order to operate the PID loop. Ramp/Soak will be discussed in another section.



Complete the PID Setup

Once you have filled in the necessary information for the basic PID setup, the configuration should be saved. The icons on the Setup PID dialog will allow you to save the configuration to the PLC and to disk. The save to icons have the arrow pointing to the PLC and disk. The read from icons have the arrows pointing away from the PLC and disk.

An optional feature is available with the Doc tab in the Setup PID window. You enter a name and description for the loop. This is useful if there is more than one PID loop in your application.



NOTE: *It is good practice to save your project after setting up the PID loop by selecting* ***File*** *from the menu toolbar, then* ***Save project > to disk****. In addition to saving your entire project, all the PID parameters are also saved.*

PID Loop Tuning

Once you have set up a PID loop, it must be tuned in order for it to work. The goal of loop tuning is to adjust the loop gains so the loop has optimal performance in dynamic conditions. The quality of a loop's performance may generally be judged by how well the PV follows the SP after an SP step change. It is important to keep in mind that understanding the process is fundamental to getting a well designed control loop. Sensors must be in appropriate locations and valves must be sized correctly with appropriate trim. **PID control does not have *typical* values**. There isn't one control process that is identical to another.

Manual Tuning vs. Auto Tuning

You may enter the PID gain values to tune your loops (manual tuning), or you can rely on the PID processing engine in the CPU to automatically calculate the gain values (auto tuning). Most experienced process engineers will have a favorite method; the DL205 will accommodate either preference. The use of auto tuning can eliminate much of the trial-and-error of the manual tuning approach, especially if you do not have a lot of loop tuning experience. However, performing the auto tuning procedure will get the gains close to optimal values, but additional manual tuning can get the gain values to their optimal values.

WARNING: Only authorized personnel fully familiar with all aspects of the process should make changes that affect the loop tuning constants. Using the loop auto tune procedures will affect the process, including inducing large changes in the control output value. Make sure you thoroughly consider the impact of any changes to minimize the risk of injury to personnel or damage to equipment. The auto tune in the DL205 is not intended to be used as a replacement for your process knowledge.

Open-Loop Test

Whether you use manual or auto tuning, it is very important to verify basic characteristics of a newly-installed process before attempting to tune it. With the loop in Manual Mode, verify the following items for each new loop.

- **Setpoint** – verify that the SP source can generate a setpoint. Put the PLC in Run Mode and leave the loop in Manual Mode, then monitor the loop table location V+02 to see the SP value(s). (If you are using the ramp/soak generator, test it now.)
- **Process Variable** – verify that the PV value is an accurate measurement, and the PV data arriving in the loop table location V+03 is correct. If the PV signal is very noisy, consider filtering the input either through hardware (RC low-pass filter), or using the filter in this chapter.
- **Control Output** – if it is safe to do so, manually change the output a small amount (perhaps 10%) and observe its affect on the process variable. Verify the process is direct-acting or reverse acting, and check the setting for the control output (inverted or non-inverted). Make sure the control output upper and lower limits are not equal to each other.
- **Sample Rate** – while operating open-loop, this is a good time to find the ideal sample rate (see Configure the PID Loop on page 8-25). However, if you are going to use auto tuning, the auto tuning procedure will automatically calculate the sample rate in addition to the PID gains.

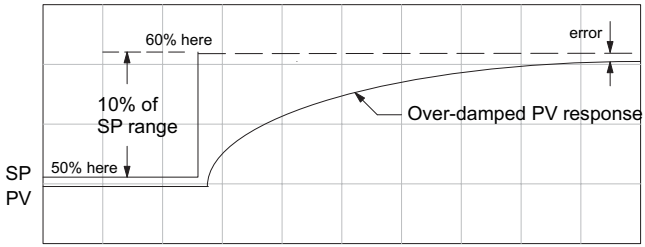
Manual Tuning Procedure

It is not necessary to try to obtain the best values for the P, I and D parameters in the PID loop by trial and error. Following is a typical procedure for tuning a temperature control loop, which you may use to tune your loop.

Monitor the values of SP, PV and CV with a loop trending instrument or use the PID View feature in *Direct*SOFT (see page 8-49).

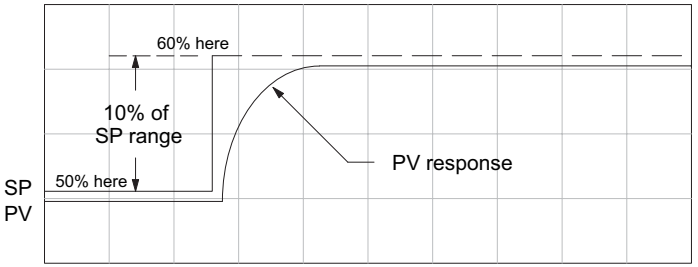
NOTE: *We recommend using the PID View Tuning and Trending window to select manual for the vertical scale feature, for both SP/PV area and Bias/Control Output areas. The auto scaling feature would otherwise change the vertical scale on the process parameters and add confusion to the loop tuning process.*

- Adjust the gains so the Proportional Gain = 0.5 or 1.0 (1.0 is a good value based on experience), Integral Gain = 9999 (this basically eliminates reset) and Derivative Gain = 0000. This disables the integrator and derivative terms, and provides some proportional gain.
- Check the bias value in the PID View and set it to zero.
- Set the SP to a value equal to 50% of the full range.
- Now, select Auto Mode. If the loop will not stay in Auto Mode, check the troubleshooting tips at the end of this chapter. Allow the PV to stabilize around the 50% point of the range.
- Change the SP to the 60% point of the range.

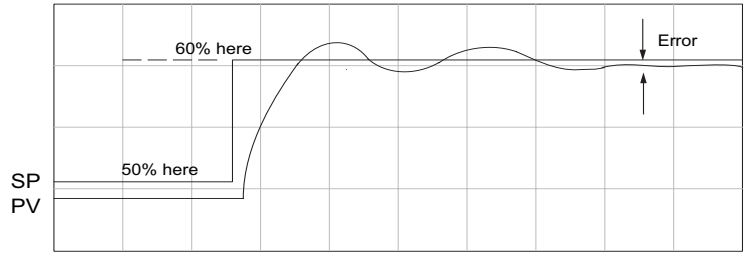


The response may take awhile, but you will see that there isn't any oscillation. This response is not desirable since it takes a long time to correct the error; also, there is a difference between the SP and the PV.

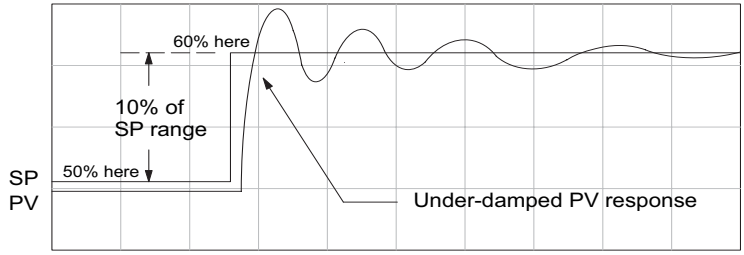
- Increase the Proportional gain, for example to 2.0. The control output will be greater and the response time will be quicker. The trend should resemble the figure below.



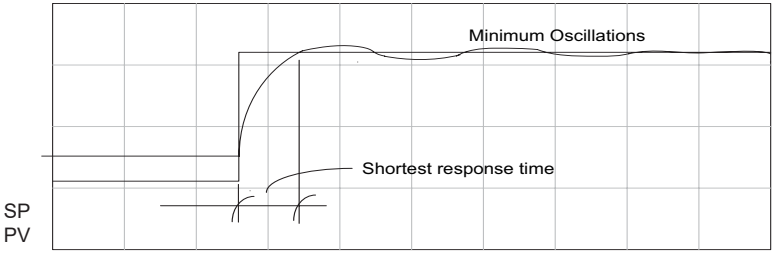
- Increase the Proportional gain in small increments, such as 4, 6, 7, etc., until the control output response begins to oscillate. This is the Proportional gain that should be recorded.
- Now, return the Proportional gain to the stable response; for example, 9.7. The error, SP-PV, should be small, but not at zero.



- Next, add a small amount of Integral gain (reset) in order for the error to reach zero. Begin by using 80 seconds (adjust in minutes if necessary). The error should get smaller.
- Set the Integral gain to a lower value, such as 50 for a different response. If there is no response, continue to decrease the reset value until the response becomes unstable. See the figure below.



- For discussion, let us say that a reset value of 35 made the control output unstable. Return the reset value to the stable value, such as 38. **Be careful with this adjustment since the oscillation can destroy the process.**
- The control output response should be optimal now, without a Derivative gain. The example recorded values are: Proportional gain = 9.7 and Integral gain = 38 seconds. Note that the error has been minimized.



The manual method is the most common method used to tune a PID loop. Derivative gain is almost never used in a temperature control loop. This method can also be used for other control loops, but other parameters may need to be added for a stable control output.

Test your loop for a high PV of 80% and again for a low PV of 20%, and correct the values if necessary. Small adjustments of the parameters can make the control output more precise or more unstable. It is sometimes acceptable to have a small overshoot to make the control output react quicker. The derivative gain can be helpful for those control loops which are not controlling temperature. For these loops, try adding a value of 0.5 for the derivative gain and see if this improves the control output. If there is little or no response, increase the derivative by increments of 0.5 until there is an improvement to the output trend. Recall that the derivative gain reacts with a rate of change of the error.

Alternative Manual Tuning Procedures by Others

The following tuning procedures have been extracted from various publications about PID process control. These procedures are for comparison to the procedure in this manual.

Tuning PID Controllers

Two-Mode Simple Method – for P-I controllers

1. Turn off reset and set the gain to a small value (0.5–1.0).
2. Increase gain until cycling starts, then decrease gain slightly.
3. Make setpoint changes to observe offset (error).
4. Increase reset to eliminate offset (error).
5. Repeat steps 2 through 4 until you obtain the largest gain and reset consistent with the criteria of the control desired, i.e., offset, overshoot, stability.

Zeigler-Nichols Method – "Quarter amplitude decay"

1. Turn off reset and rate and set the proportional gain to a fairly large value.
2. Make a small setpoint change and observe how the controlled variable cycles.
3. Adjust the gain until the cycle is self-sustaining, and of constant amplitude; this value is the ultimate gain (G_u).
4. Measure the period of cycling in minutes. This is the ultimate period (P_u).
5. Calculate the controller adjustments as follows:

P only:	$G = G_u/2$	
P & I :	$G = G_u/2.2$	
	$Ti = 1.2/P_u$	(repeats/minute)
P-I-D:	$G = G_u/1.6$	
	$Ti = 2.0/P_u$	(repeats/minute)
	$Td = P_u/8.0$	(minutes)

Pessen Method

1. Follow the procedure described above (Zeigler-Nichols) to determine the ultimate gain and ultimate period.
2. Apply the formulas below.

•For no overshoot during startup:

$G = G_u/5.0$
$Ti = 3/P_u$ (repeats/minute)
$Td = Pu/2$ (minutes)

•For some overshoot, but better response to disturbances:

$G = G_u/3$
$Ti = 3/P_u$ (repeats/minute)
$Td = P_u/3$ (minutes)

Auto Tuning Procedure

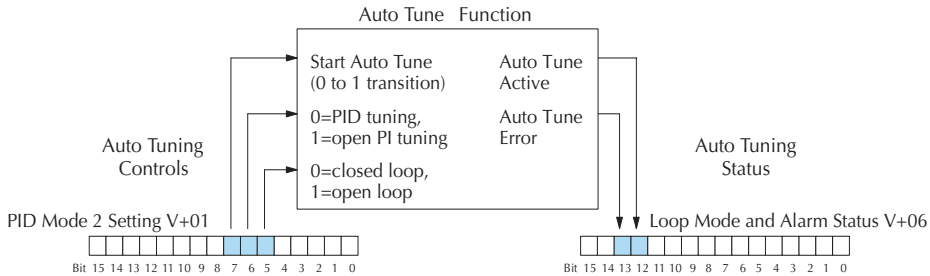
The auto tuning feature for the DL205 loop controller will only run once each time it is enabled in the PID table. Therefore, auto tuning does not run continuously during operation (this would be *adaptive* control). Whenever there is a substantial change in loop dynamics, such as mass of process, size of actuator, etc., the tuning process will need to be repeated in order to derive new gains required for optimal control.

WARNING: Only authorized personnel fully familiar with all aspects of the process should make changes that affect the loop tuning constants. Using the loop auto tuning procedures will affect the process, including inducing large changes in the control output value. Make sure you thoroughly consider the impact of any changes to minimize the risk of injury to personnel or damage to equipment. The auto tune in the DL205 is not intended to be used as a replacement for your process knowledge.

Once the physical loop components are connected to the PLC, auto tuning can be initiated within *Direct*SOFT (see the *Direct*SOFT Programming Software Manual), and it can be used to establish initial PID parameter values. Auto tuning is the best "guess" the CPU can do after some trial tests.

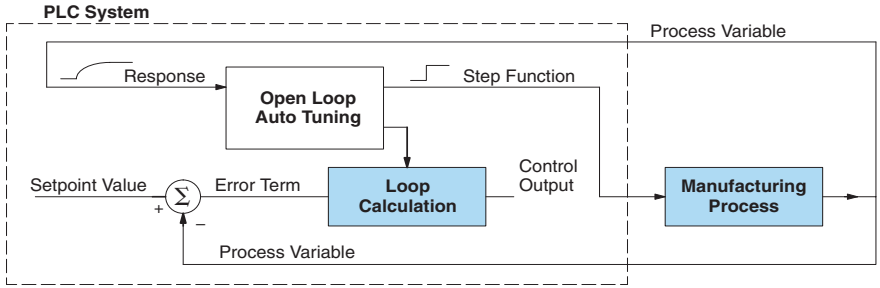
The loop controller offers both closed-loop and open-loop methods. The following sections describe how to use the auto tuning feature, and what occurs in open and closed-loop auto tuning.

The controls for the auto tuning function use three bits in the PID Mode Setting 2 word V+01, as shown below. *Direct*SOFT will manipulate these bits automatically when you use the auto tune feature within *Direct*SOFT. Or, you may have your ladder logic access these bits directly for allowing control from another source such as a dedicated operator interface. The individual control bits allow you to start the auto tune procedure, select PID or PI tuning and select closed-loop or open-loop tuning. If you select PI tuning, the auto tune procedure leaves the derivative gain at 0. The Loop Mode and Alarm Status word V+06 reports the auto tune status as shown. Bit 12 will be on (1) during the auto tune cycle, automatically returning to off (0) when done.



Open-Loop Auto Tuning

During an open-loop auto tuning cycle, the loop controller operates as shown in the diagram below. Before starting this procedure, place the loop in Manual Mode and ensure the PV and control output values are in the middle of their ranges (away from the end points).

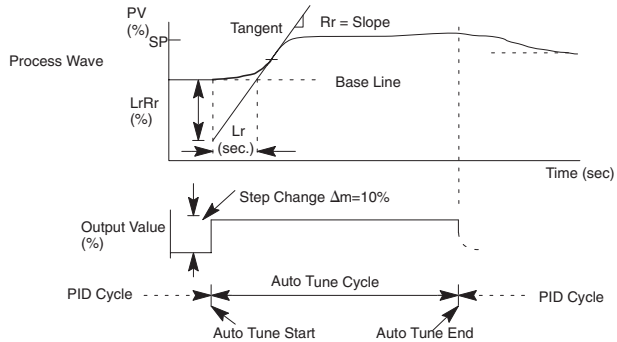


NOTE: *In theory, the SP value does not matter in this case, because the loop is not closed. However, the requirement of the firmware is that the SP value must be more than 5% of the PV range from the actual PV before starting the auto tune cycle (for the DL205, 12 bit PV should be 205 counts or more below the SP for forward-acting loops, or 205 counts or more above the SP for reverse-acting loops).*

When auto tuning, the loop controller induces a step change on the output and simply observes the response of the PV. From the PV response, the auto tune function calculates the gains and the sample time. It automatically places the results in the corresponding registers in the loop table.

The following timing diagram shows the events which occur in the open-loop auto tuning cycle. The auto tune function takes control of the control output and induces a 10%-of-span step change. If the PV change which the loop controller observes is less than 2%, then the step change on the output is increased to 20%-of-span.

- When Auto Tune starts, step change output m=10%
- During Auto Tune, the controller output reached the full scale positive limit. Auto Tune stopped and the Auto Tune Error bit in the Alarm word bit turned on.
- When PV change is under 2%, output is changed at 20%. Open Loop Auto Tune Cycle Wave: Step Response Method



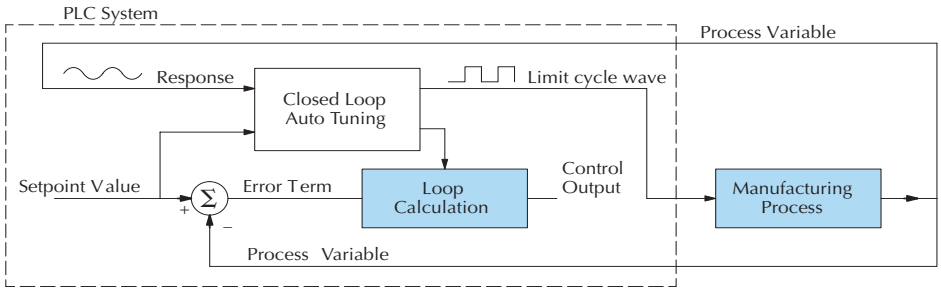
When the loop tuning observations are complete, the loop controller computes Rr (maximum slope in %/sec.) and Lr (dead time in sec). The auto tune function computes the gains according to the Zeigler-Nichols equations, shown below:

PID Tuning	PI Tuning
P=1.2*Δm/LrRr	P=0.9*Δm/LrRr
I=2.0* Lr	I=3.33* Lr
D=0.5* Lr	D=0
Sample Rate = 0.056* Lr	Sample Rate = 0.12*Lr
Δm = Output step change (10% = 0.1, 20% = 0.2)	

We highly recommend using *Direct*SOFT for the auto tuning interface. The duration of each auto tuning cycle will depend on the mass of the process. A slowly-changing PV will result in a longer auto tune cycle time. When the auto tuning is complete, the proportional, integral, and derivative gain values are automatically updated in loop table locations V+10, V+11, and V+12 respectively. The sample time in V+07 is also updated automatically. You can test the validity of the values the auto tuning procedure yields by measuring the closed-loop response of the PV to a step change in the output. The instructions on how to do this are in the section on the manual tuning procedure (located prior to this auto tuning section).

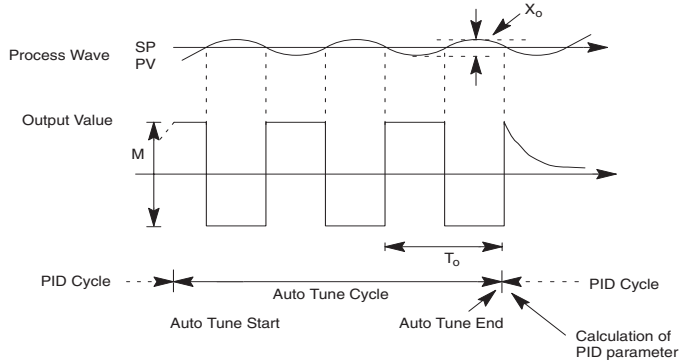
Closed-Loop Auto Tuning

During a closed-loop auto tuning cycle, the loop controller operates as shown in the illustration below.



When auto tuning, the loop controller imposes a square wave on the output. Each transition of the output occurs when the PV value crosses over/under the SP value. Therefore, the frequency of the limit cycle is roughly proportional to the mass of the process. From the PV response, the auto tune function calculates the gains and the sample time. It automatically places the results in the corresponding registers in the loop table.

The following timing diagram shows the events which occur in the closed-loop auto tuning cycle. The auto tune function examines the direction of the offset of the PV from the SP. The auto tune function then takes control of the control output and induces a full-span step change in the opposite direction. Each time the sign of the error (SP – PV) changes, the output changes full-span in the opposite direction. This proceeds through three full cycles.



*Mmax = Output Value upper limit setting. Mmin = Output Value lower limit setting.

* This example is direct–acting.

When set to reverse–acting, the output will be inverted. When the loop tuning observations are complete, the loop controller computes To (bump period) and Xo (amplitude of the PV). Then it uses these values to compute Kpc (sensitive limit) and Tpc (period limit). From these values, the loop controller auto tune function computes the PID gains and the sample rate according to the Zeigler-Nichols equations shown below:

$$Kpc = 4M / (\pi * X0) \qquad Tpc = 0$$

M = Amplitude of output

PID Tuning	PI Tuning
P = 0.45*Kpc	P = 0.30*Kpc
I = 0.60*Tpc	I = 1.00*Tpc
D = 0.10*Tpc	D = 0
Sample Rate = 0.014*Tpc	Sample Rate = 0.03*Tpc

Auto tuning error

In open-loop tuning, if the auto tune error bit (bit 13 of loop Mode/Alarm status word V+06) is on, verify the PV and SP values are within 5% of full scale difference, as required by the auto tune function.

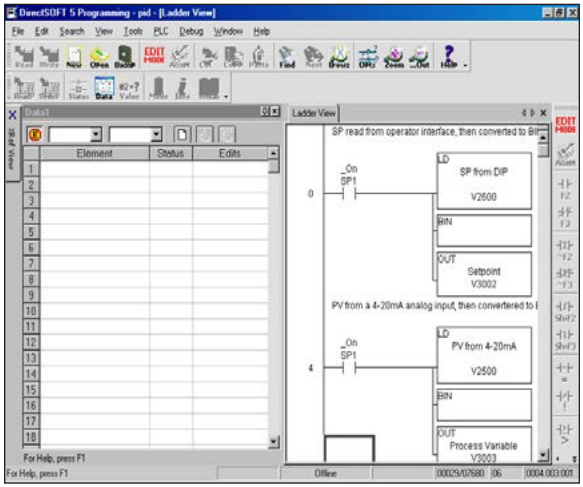
***NOTE**: If your PV fluctuates rapidly, you probably need to use the built-in analog filter (see page 8–55) or create a filter in ladder logic (see example on page 8–56).*

Use DirectSOFT Data View with PID View

The **Data View** window is a very useful tool which can be used to help tune your PID loop. You can compare the variables in the **PID View** with the actual values in the V-memory location with Data View.

Open a New Data View Window

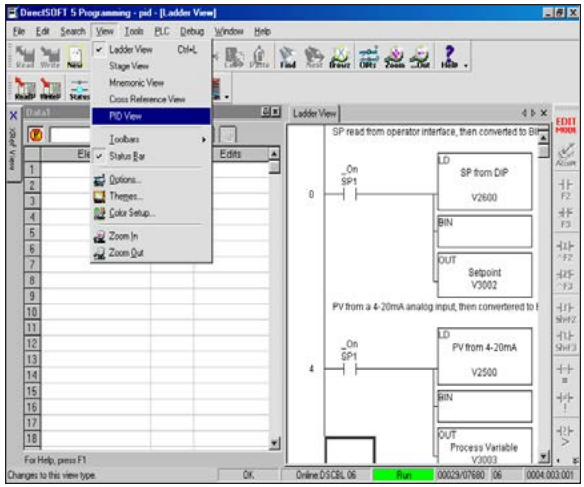
A new Data View window can be opened in any one of three ways; the menu bar **Debug > Data View > New**, the keyboard shortcut **Ctrl + Shift + F3** or the **Data** button on the Status toolbar. By default, the Data View window is assigned Data1 as the default name. This name can be changed for the current view using the Options dialog. The following diagram is an example of a newly opened Data View. The window will open next to the Ladder View by default.



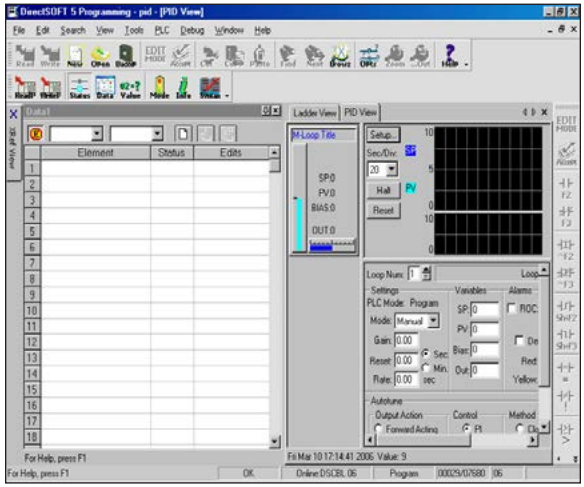
The Data View window can be used just as it is shown above for troubleshooting your PID logic, and it can be most useful when tuning the PID loop.

Open PID View

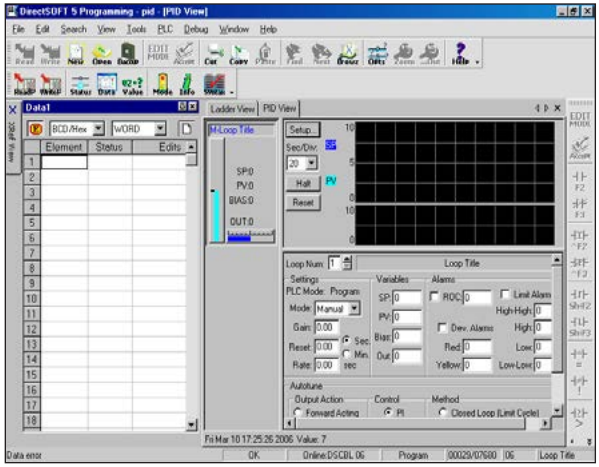
The PID View can only be opened after a loop has been setup in your ladder program. PID View is opened by selecting it from the View submenu on the Menu bar, **View > PID View**. The PID View can also be opened by clicking on the PID View button from the PLC Setup toolbar if it is in view.



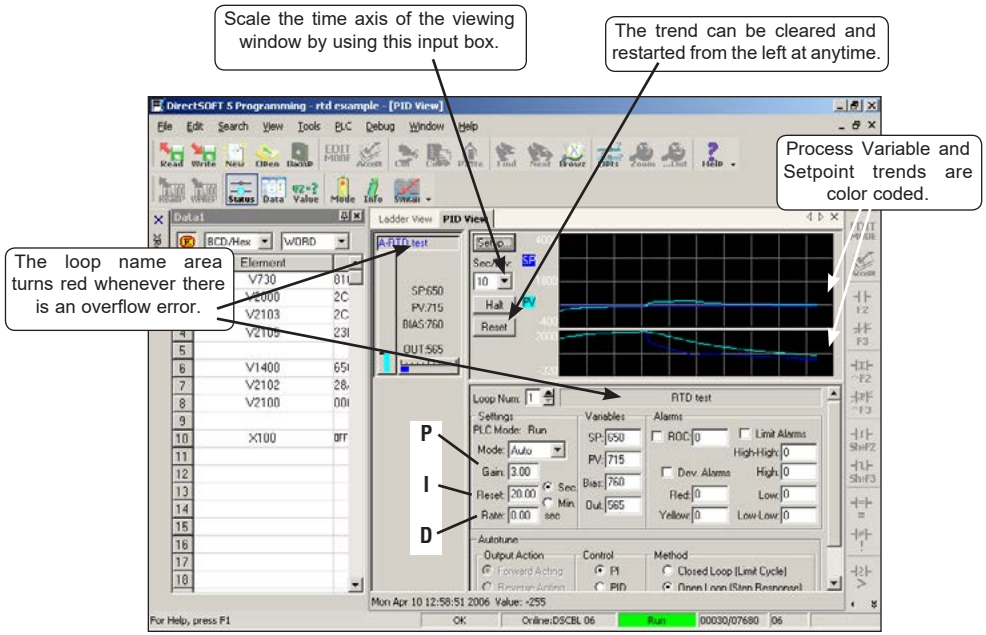
The PID View will open and appear over the Ladder View which can be brought into view by clicking on its tab. When using the Data View and the PID View together, each view can be sized for better use as shown on the facing page.



The two views are now ready to be used to tune your loop. You will be able to see where the PID values have been set and see the process that it is controlling.



With both windows positioned in this manner, you are able to see where the PID values have been set and see the process that it is controlling. In the diagram below, you can see the current SP, PV and Output values, along with the other PID addresses. Refer to the Loop Table Word Definitions (page 8-20) for details on each word in the table. This is also a good data type reference for each word in the table.

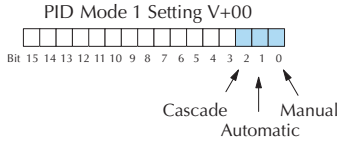


Using the Special PID Features

It's a good idea to understand the special features of the DL205 and how to use them. You may want to incorporate some of these features for your PID.

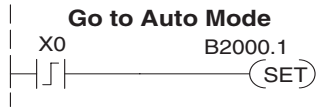
How to Change Loop Modes

The first three bits of the PID Mode Setting 1 word (V+00) request the operating mode of the corresponding loop. Note: these bits are mode change *requests*, not commands (certain conditions can prohibit a particular mode change – see next page).



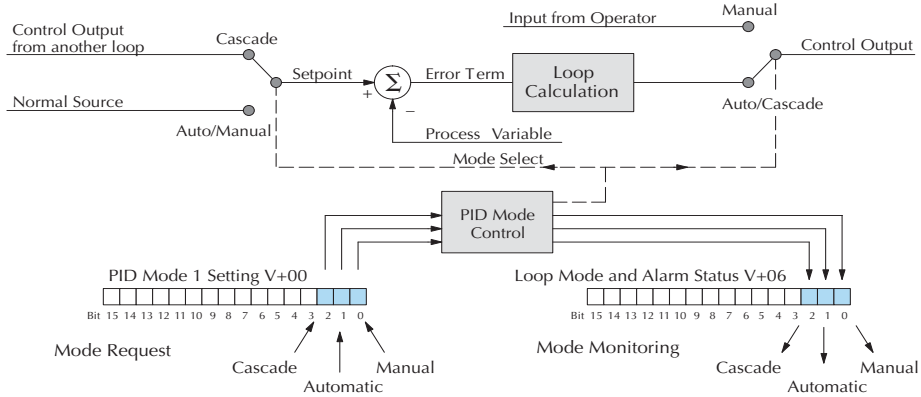
The normal state of these mode request bits is "000." To request a mode change, you must SET the corresponding bit to a "**1**" using a one-shot. The PID loop controller automatically resets the bits back to "000" after it reads the mode change request. Methods of requesting mode changes are:

- ***Direct***SOFT's **PID View** – this is the easiest method. Use the pull-down menu, or click on one of the radio buttons if using older *Direct*SOFT versions, and the appropriate bit will get set.
- **Ladder program**– ladder logic can request any loop mode when the PLC is in Run Mode. This will be necessary after application startup if mode changes are part of the application. Use the program ladder shown to the right to SET the mode bit (do not use an OUT coil). On a 0–1 transition of X0, the rung sets the Auto bit equal to 1. The loop controller resets it.



- **Operator panel** – interface the operator's panel to ladder logic using standard methods, then use the logic to the right to set the mode bit.

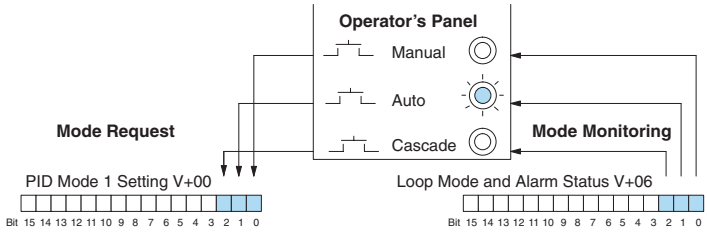
Since mode changes can only be *requested*, the PID loop controller will decide when to permit mode changes and provide the loop mode status. It reports the current mode on bits 0, 1, and 2 of the Loop Mode/Alarm Status word, location V+06 in the loop table. The parallel request/monitoring functions are shown in the figure below. The figure also shows the two possible mode-dependent SP sources, and the two possible Control Output sources.



Operator Panel Control of PID Modes

Since the modes Manual, Auto and Cascade are the most fundamental and important PID loop controls, you may want to hard-wire mode control switches to an operator's panel. Most applications will need only Manual and Auto selections (Cascade is used in special applications). Remember that mode controls are really *mode request* bits, and the actual loop mode is indicated elsewhere.

The following figure shows an operator's panel using momentary push-buttons to request PID mode changes. The panel's mode indicators do not connect to the switches, but interface to the corresponding data locations.



PLC Modes Effect on Loop Modes

If you have selected the option for the loops to follow the PLC mode, the PLC modes (Program, Run) interact with the loops as a group. The following summarizes this interaction:

- When the PLC is in Program Mode, all loops are placed in Manual Mode and no loop calculations occur. However, note that output modules (including analog outputs) turn off in PLC Program Mode. So, actual manual control is not possible when the PLC is in Program Mode.
- The only time the CPU will allow a loop mode change is during PLC Run Mode operation. As such, the CPU records the modes of all four loops as the desired mode of operation. If power failure and restoration occurs during PLC Run Mode, the CPU returns all loops to their prior mode (which could be Manual, Auto, or Cascade).
- On a Program-to-Run mode transition, the CPU forces each loop to return to its prior mode recorded during the last PLC Run Mode.
- You can add and configure new loops only when the PLC is in Program Mode. New loops automatically begin in Manual Mode.

Loop Mode Override

In normal conditions, the mode of a loop is determined by the request to V+00, bits 0, 1, and 2. However, some conditions exist which will prevent a requested mode change from occurring:

- A loop that is not set independent of PLC mode cannot change modes when the PLC is in Program mode.
- A major loop of a cascaded pair of loops cannot go from Manual to Auto until its minor loop is in Cascade mode.

In other situations, the PID loop controller will automatically change the mode of the loop to ensure safe operation:

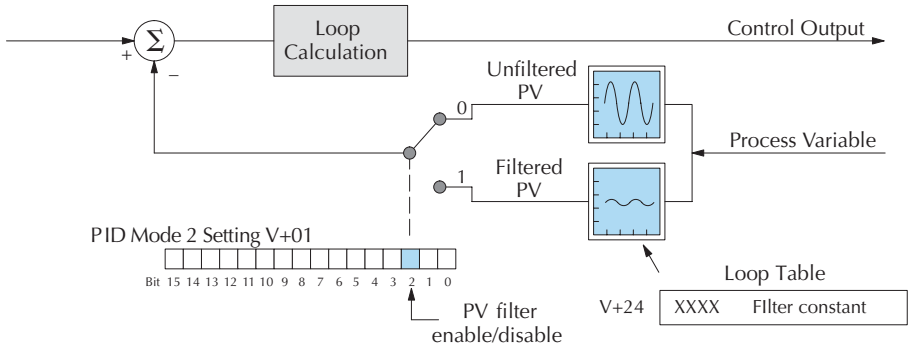
- A loop which develops an error condition automatically goes to Manual.
- If the minor loop of a cascaded pair of loops leaves Cascade Mode for any reason, its major loop automatically goes to Manual Mode.

PV Analog Filter

A noisy PV signal can make tuning difficult and can cause the control output to be more extreme than necessary, as the output tries to respond to the peaks and valleys of the PV. There are two equivalent methods of filtering the PV input to make the loop more stable. The first method is accomplished using the DL205's built-in filter. The second method achieves a similar result using ladder logic.

The DL205 Built-in Analog Filter

The DL205 provides a selectable first-order low-pass PV input filter. **We only recommend the use of a filter during auto tuning or PID control if there is noise on the input signal.** You may disable the filter after auto tuning is complete, or continue to use it if the PV input signal is noisy.



Bit 2 of PID Mode Setting 2 provides the enable/disable control for the low-pass PV filter (0=disable, 1=enable). The roll-off frequency of the single-pole low-pass filter is controlled by using register V+24 in the loop parameter table, the filter constant. The data format of the filter constant value is BCD, with an implied decimal point 00X.X, as follows:

- The filter constant has a valid range of 000.1 to 001.0. **The smaller the filter value, the greater the filtering performed; for example, the value 001.0 provides no filtering.**
- ***Direct***SOFT converts values above the valid range to 001.0 and values below this range to 000.1.
- Values close to 001.0 result in higher roll-off frequencies, while values closer to 000.1 result in lower roll-off frequencies.

We highly recommend using ***Direct***SOFT for the auto tuning interface. The duration of each auto tuning cycle will depend on the mass of your process. A slowly-changing PV will result in a longer auto tune cycle time.

When the auto tuning is complete, the proportional and integral gain values are automatically updated in loop table locations V+10 and V+11 respectively. The derivative is calculated if you autotune for PID and updated in loop table location V+12. The sample time in V+07 is also updated automatically. You can test the validity of the values the auto tuning procedure yields by measuring the closed-loop response of the PV to a step change in the output. The instructions on how to do this are in the section on the manual tuning procedure.

The algorithm that the built-in filter follows is:

$$y_i = k\,(x_i - y_i{-}1) + y_i{-}1$$

where:

y_i is the current output of the filter

x_i is the current input to the filter

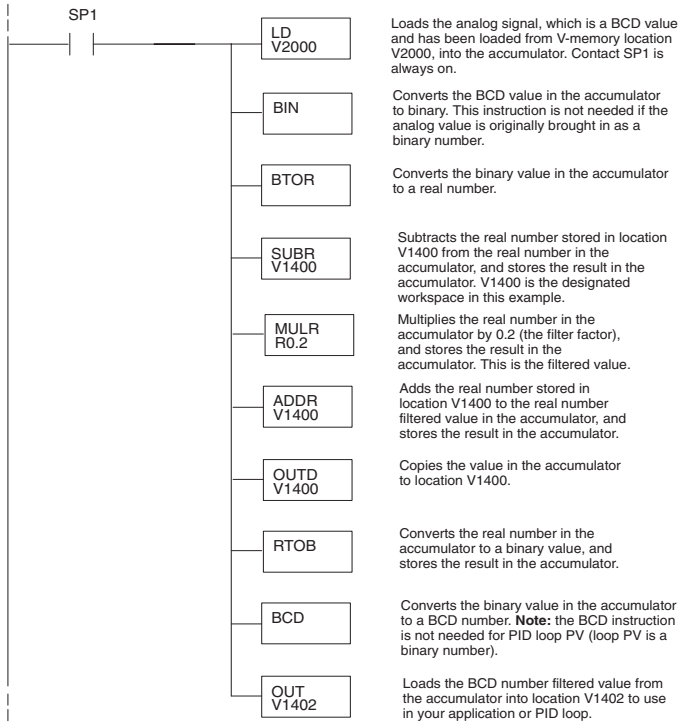
y_i–1 is the previous output of the filter

k is the PV Analog Input Filter Factor

Creating an Analog Filter in Ladder Logic

A similar algorithm can be built in your ladder program. Your analog inputs can be filtered effectively using either method. The following programming example describes the ladder logic you will need. Be sure to change the example memory locations to those that fit your application.

Filtering can induce a 1 part in 1000 error in your output because of "rounding." If your process cannot tolerate a 1 part in 1000 error, do not use filtering. Because of the rounding error, you should not use zero or full scale as alarm points. Additionally, the smaller the filter constant the greater the smoothing effect, but the slower the response time. Be sure a slower response is acceptable in controlling your process.



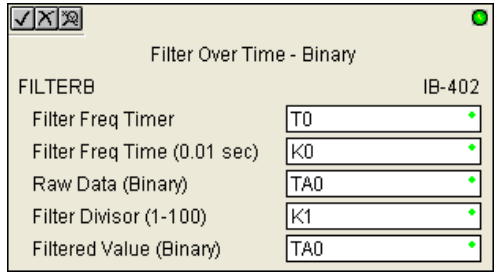
Use the DirectSOFT Filter Intelligent Box Instruction

For those who are using DirectSOFT, you have the opportunity to use Intelligent Box (IBox) instruction IB-402, Filter Over Time in Binary (decimal). This IBox will perform a first-order filter on the Raw Data on a defined time interval. The equation is,

New = Old + [(Raw - Old) / FDC] where

- New =New Filtered Value
- Old = Old Filtered Value
- FDC = Filter Divisor Constant
- Raw = Raw Data

The Filter Divisor Constant is an integer in the range K1 to K100, such that if it equaled K1, then no filtering is performed.

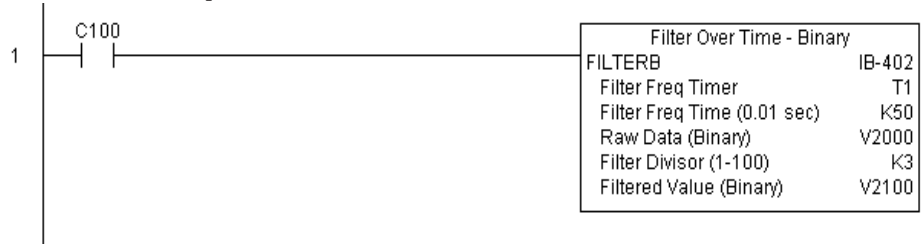


The rate at which the calculation is performed is specified by time in hundredths of a second (0.01 seconds) as the Filter Freq Time parameter. Note that this Timer instruction is embedded in the IBox and must NOT be used any other place in your program. Power flow controls whether the calculation is enabled. If it is disabled, the Filter Value is not updated. On the first scan from Program to Run mode, the Filter Value is initialized to 0 to give the calculation a consistent starting point.

Since the following binary filter example does not write directly to the PID PV location, the BCD filter could be used with BCD values and then converted to BIN.

FilterB Example

Following is an example of how the FILTERB IBox is used in a ladder program. The instruction is used to filter a binary value that is in V2000. Timer (T1) is set to 0.5 seconds, the rate at which the filter calculation will be performed. The filter constant is set to 3.0. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering. The filtered value will be placed in V2100.

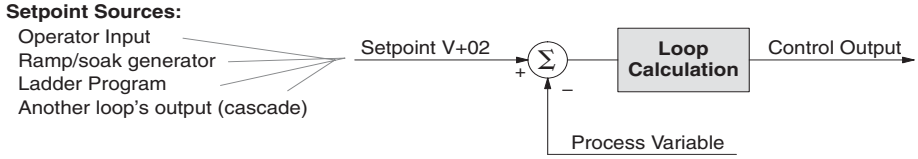


NOTE: *See Chapter 5 of this manual for more detailed information.*

Ramp/Soak Generator

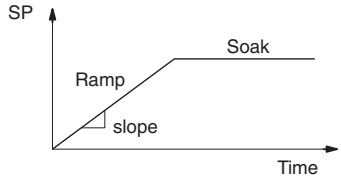
Introduction

Our discussion of basic loop operation noted the setpoint for a loop will be generated in various ways, depending on the loop operating mode and programming preferences. In the figure below, the ramp/soak generator is one of the ways the SP may be generated. It is the responsibility of your ladder program to ensure only one source attempts to write the SP value at V+02 at any particular time.



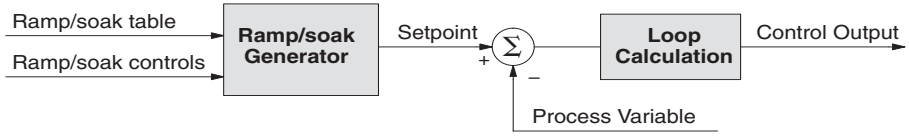
If the SP for your process rarely changes or can tolerate step changes, you probably will not need to use the ramp/soak generator. However, some processes require precisely-controlled SP value changes. *The ramp/soak generator can greatly reduce the amount of programming required for these applications.*

The terms **ramp** and **soak** have special meanings in the process control industry, and refer to desired setpoint (SP) values in temperature control applications. In the figure to the right, the setpoint increases during the ramp segment. It remains steady at one value during the soak segment.



Complex SP profiles can be generated by specifying a series of ramp/soak segments. The ramp segments are specified in SP units per second time. The soak time is also programmable in minutes.

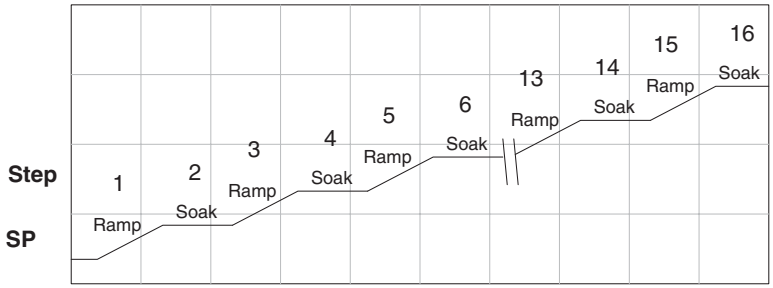
It is instructive to view the ramp/soak generator as a dedicated function to generate SP values, as shown below. It has two categories of inputs that determine the SP values generated. The ramp/soak table must be programmed in advance, containing the values that will define the ramp/soak profile. The loop reads from the table during each PID calculation as necessary. The ramp/soak controls are bits in a special loop table word that control the real-time start/stop functionality of the ramp/soak generator. The ladder program can monitor the status of the ramp soak profile (current ramp/segment number).



Now that we have described the general ramp/soak generator operation, we list its specific features:

- Each loop has its own ramp/soak generator (use is optional).
- You may specify up to eight ramp/soak steps (16 segments).
- The ramp/soak generator can run any time the PLC is in Run mode. Its operation is independent of the loop mode (Manual or Auto).
- Ramp/soak real-time controls include Start, Hold, Resume, and Jog.
- Ramp/soak monitoring includes Profile Complete, Soak Deviation (SP minus PV), and current ramp/soak step number.

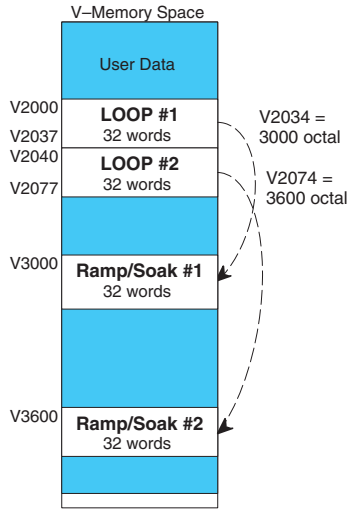
The following figure shows an SP profile consisting of ramp/soak segment pairs. The segments are individually numbered as steps from 1 to 16. The slope of each of the ramp segments may be either increasing or decreasing. The ramp/soak generator automatically knows whether to increase or decrease the SP based on the relative values of a ramp's end points. These values come from the ramp/soak table.



Ramp/Soak Table

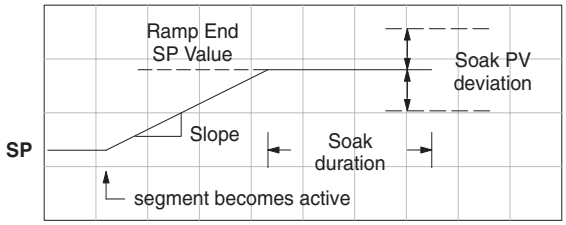
The parameters which define the ramp/soak profile for a loop are in a ramp/soak table. Each loop may have its own ramp/soak table, but it is optional. Recall the Loop Parameter table consists of a 32-word block of memory for each loop, and together they occupy one contiguous memory area. However, the ramp/soak table for a loop is individually located, because it is optional for each loop. An address pointer in location V+34 in the loop table specifies the starting location of the ramp/soak table.

In the example to the right, the loop parameter tables for Loop #1 and #2 occupy contiguous 32-word blocks as shown. Each has a pointer to its ramp/soak table, independently located elsewhere in user V-memory. Of course, you may locate all the tables in one group, as long as they do not overlap.



The parameters in the ramp/soak table must be user-defined. The most convenient way is to use *Direct*SOFT, which features a special editor for this table. Four parameters are required to define a ramp and soak segment pair, as pictured below.

- Ramp End Value – specifies the destination SP value for the end of the ramp. Use the same data format for this number as you use for the SP. It may be above or below the beginning SP value, so the slope could be up or down (we don't have to know the starting SP value for ramp #1).
- Ramp Slope – specifies the SP increase in counts (units) per second. It is a BCD number from 00.00 to 99.99 (uses implied decimal point).
- Soak Duration – specifies the time for the soak segment in minutes, ranging from 000.1 to 999.9 minutes in BCD (implied decimal point).
- Soak PV Deviation – (optional) specifies an allowable PV deviation above and below the SP value during the soak period. A PV deviation alarm status bit is generated by the ramp/soak generator.



Ramp/Soak Table

V+00	XXXX	Ramp End SP Value
V+01	XXXX	Ramp Slope
V+02	XXXX	Soak Duration
V+03	XXXX	Soak PV Deviation

The ramp segment becomes active when the previous soak segment ends. If the ramp is the first segment, it becomes active when the ramp/soak generator is started, and automatically assumes the present SP as the starting SP.

Offset	Step	Description	Offset	Step	Description
+ 00	1	Ramp End SP Value	+ 20	9	Ramp End SP Value
+ 01	1	Ramp Slope	+ 21	9	Ramp Slope
+ 02	2	Soak Duration	+ 22	10	Soak Duration
+ 03	2	Soak PV Deviation	+ 23	10	Soak PV Deviation
+ 04	3	Ramp End SP Value	+ 24	11	Ramp End SP Value
+ 05	3	Ramp Slope	+ 25	11	Ramp Slope
+ 06	4	Soak Duration	+ 26	12	Soak Duration
+ 07	4	Soak PV Deviation	+ 27	12	Soak PV Deviation
+ 10	5	Ramp End SP Value	+ 30	13	Ramp End SP Value
+ 11	5	Ramp Slope	+ 31	13	Ramp Slope
+ 12	6	Soak Duration	+ 32	14	Soak Duration
+ 13	6	Soak PV Deviation	+ 33	14	Soak PV Deviation
+ 14	7	Ramp End SP Value	+ 34	15	Ramp End SP Value
+ 15	7	Ramp Slope	+ 35	15	Ramp Slope
+ 16	8	Soak Duration	+ 36	16	Soak Duration
+ 17	8	Soak PV Deviation	+ 37	16	Soak PV Deviation

Many applications do not require all 16 ramp/soak steps. Use all zeros in the table for unused steps. The ramp/soak generator ends the profile when it finds ramp slope = 0.

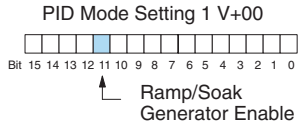
Ramp/Soak Table Flags

The individual bit definitions of the Ramp/Soak Table Flag (Addr+33) word is listed in the following table.

Bit	Ramp/Soak Flag Bit Description	Read/Write	Bit=0	Bit=1
0	Start Ramp / Soak Profile	Write	–	01 Start
1	Hold Ramp / Soak Profile		–	01 Hold
2	Resume Ramp / Soak Profile		–	01 Resume
3	Jog Ramp / Soak Profile		–	01 Jog
4	Ramp / Soak Profile Complete	Read	–	Complete
5	PV Input Ramp / Soak Deviation		Off	On
6	Ramp / Soak Profile in Hold		Off	On
7	Reserved		Off	On
8–15	Current Step in R/S Profile		decode as byte (hex)	

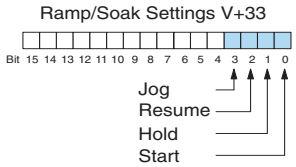
Ramp/Soak Generator Enable

The main enable control to permit ramp/soak generation of the SP value is accomplished with bit 11 in the PID Mode Setting 1 V+00 word, as shown to the right. The other ramp/soak controls in V+33 shown in the table above will not operate unless this bit=1 during the entire ramp/soak process.



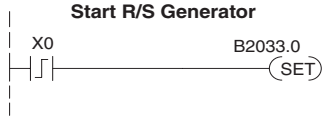
Ramp/Soak Controls

The four main controls for the ramp/soak generator are in bits 0 to 3 of the ramp/soak settings word in the loop parameter table. *Direct*SOFT controls these bits directly from the ramp/soak settings dialog. However, you must use ladder logic to control these bits during program execution. We recommend using the bit-of-word instructions.



Ladder logic must set a control bit to a 1 to command the corresponding function. When the loop controller reads the ramp/soak value, it automatically turns off the bit for you. Therefore, a reset of the bit is not required when the CPU is in Run Mode.

The example program rung to the right shows how an external switch X0 can turn on, and the PD contact uses the leading edge to set the proper control bit to start the ramp/soak profile. This uses the Set Bit-of-word instruction.

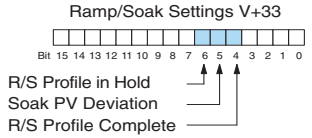


The normal state for the ramp/soak control bits is all zeros. Ladder logic must set only one control bit at a time.

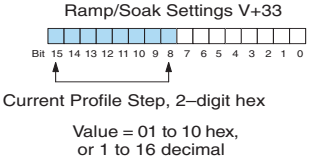
- Start – a 0 to 1 transition will start the ramp/soak profile. The CPU must be in Run Mode, and the loop can be in Manual or Auto Mode. If the profile is not interrupted by a Hold or Jog command, it finishes normally.
- Hold – a 0 to 1 transition will stop the ramp/soak profile in its current state, and the SP value will be frozen.
- Resume – a 0 to 1 transition causes the ramp/soak generator to resume operation if it is in the hold state. The SP values will resume from their previous values.
- Jog – a 0 to 1 transition will cause the ramp/soak generator to truncate the current segment (step), and go to the next segment.

Ramp/Soak Profile Monitoring

You can monitor the ramp/soak profile status using other bits in the Ramp/Soak Settings V+33 word, shown to the right.



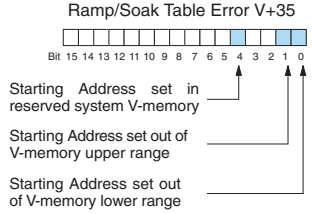
- R/S Profile Complete – =1 when the last programmed step is done.
- Soak PV Deviation – =1 when the error (SP–PV) exceeds the specified deviation in the R/S table.
- R/S Profile in Hold – =1 when the profile was active but is now in hold. Ramp/Soak Settings V+33



The number of the current step is available in the upper 8 bits of the Ramp/Soak Settings V+33 word. The bits represent a 2-digit hex number, ranging from 1 to 10. Ladder logic can monitor these to synchronize other parts of the program with the ramp/soak profile. Load this word to the accumulator and shift right 8 bits, and you have the step number.

Ramp/Soak Programming Errors

The starting address for the ramp/soak table must be a valid location. If the address points outside the range of user V-memory, one of the bits to the right will turn on when the ramp/soak generator is started. We recommend using ***Direct***SOFT to configure the ramp/soak table. It automatically range checks the addresses for you.



Testing Your Ramp/Soak Profile

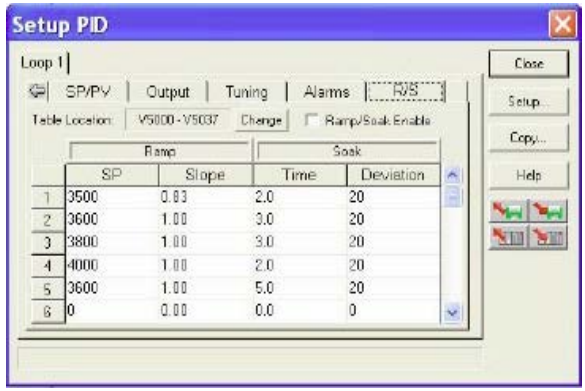
It's a good idea to test your ramp/soak profile before using it to control the process. This is easy to do, because the ramp/soak generator will run even when the loop is in Manual Mode. Using ***Direct***SOFT's PID View will be a real time-saver, because it will draw the profile on-screen for you. Be sure to set the trending timebase slow enough to display completed ramp/soak segment pairs in the waveform window.

DirectSOFT Ramp/Soak Example

The following example will step you through the Ramp/Soak setup.

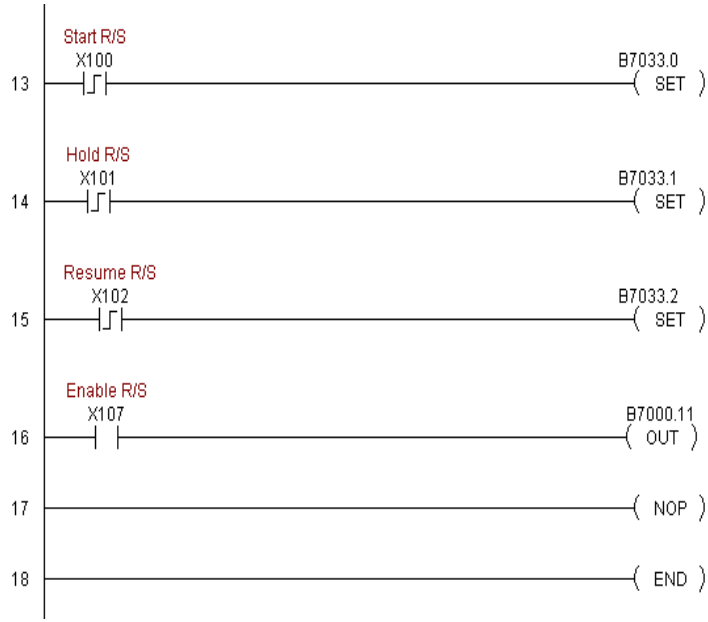
Set Up the Profile in PID Setup

The first step is to use Setup PID in *Direct*SOFT to set the profile of your process. Open the Setup PID window and select the R/S tab, and then enter the Ramp and Soak data.



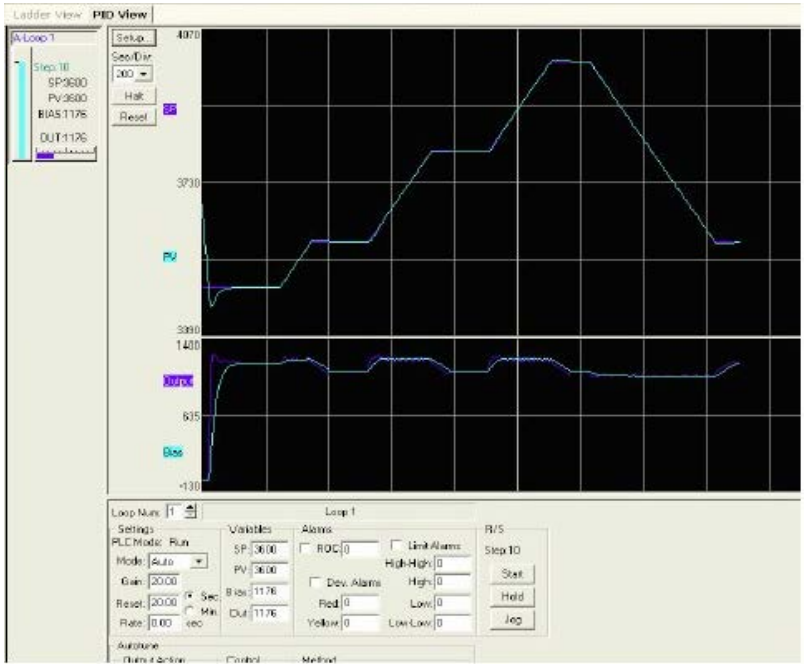
Program the Ramp/Soak Control in Relay Ladder

Refer to the Ramp/Soak Flag Bit Description table on page 8-60 when adding the control rungs to your program similar to the ladder rungs below.



Test the Profile

Test your profile using PID View.



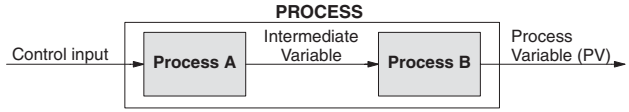
Cascade Control

Introduction

Using cascaded loops is an advanced control technique, superior to individual loop control in certain situations. As the name implies, cascade means that one loop is connected to another loop. In addition to Manual (open loop) and Auto (closed loop) Modes, the DL205 also provides Cascaded Mode.

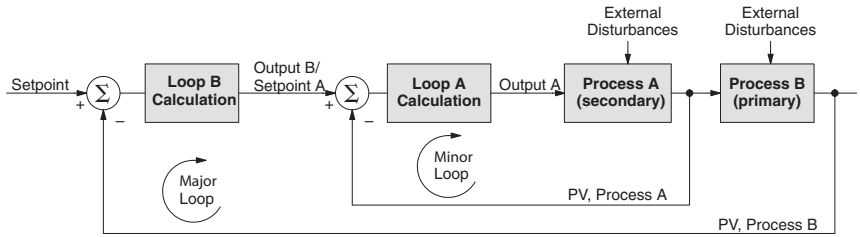
***NOTE**: Using cascaded loops is an advanced process control technique; therefore, we recommend their use only for experienced process control engineers.*

When a manufacturing process is complex and contains a lag time from control input to process variable output, even the most perfectly tuned single loop around the process may yield slow and inaccurate control. It may be the actuator operates on one physical property, which eventually affects the process variable, measured by a different physical property. Identifying the intermediate variable allows us to divide the process into two parts as shown in the following figure.



The principle of cascaded loops is simply that we add another process loop to more precisely control the intermediate variable! This separates the source of the control lag into two parts, as well.

The diagram below shows a cascade control system, showing that it is simply one loop nested inside another. The inside loop is called the minor loop, and the outside loop is called the major loop. For overall stability, the minor loop must be the fastest responding loop of the two (try a factor of 10 for a better response time). We do have to add the additional sensor to measure the intermediate variable (PV for process A). Notice the setpoint for the minor loop is automatically generated for us by using the output of the major loop. Once the cascaded control is programmed and debugged, we only need to deal with the original setpoint and process variable at the system level. The cascaded loops behave as one loop, but with improved performance over the previous single-loop solution.



One of the benefits to cascade control can be seen by examining its response to external disturbances. Remember the minor loop is faster acting than the major loop. Therefore, if a disturbance affects process A in the minor loop, the Loop A PID calculation can correct the resulting error before the major loop sees the effect.

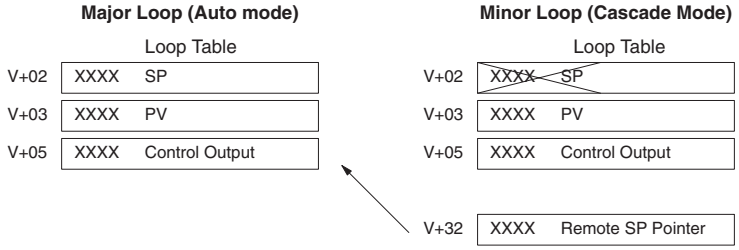
Cascaded Loops in the DL205 CPU

In using of the term cascaded loops, we must make an important distinction. Only the minor loop will actually be in the Cascade Mode. In normal operation, the major loop must be in Auto Mode. If you have more than two loops cascaded together, the outermost (major) loop must be in Auto Mode during normal operation, and all inner loops in Cascade Mode.

***NOTE**: Technically, both major and minor loops are cascaded in strict process control terminology. Unfortunately, we are unable to retain this convention when controlling loop modes. Remember that all minor loops will be in Cascade Mode, and only the outermost (major) loop will be in Auto Mode.*

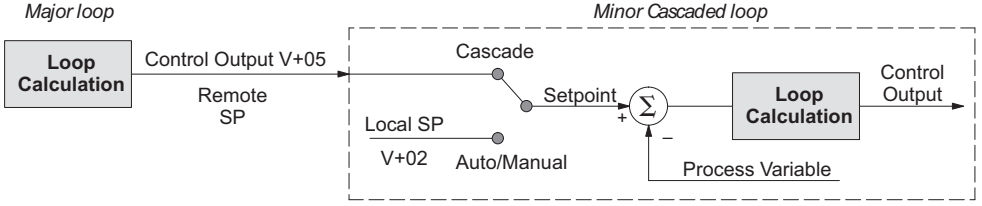
You can cascade together as many loops as necessary on the DL205, and you may have multiple groups of cascaded loops. For proper operation on cascaded loops, you must use the same data range (12/15 bit) and unipolar/bipolar settings on the major and minor loops.

To prepare a loop for Cascade Mode operation as a minor loop, you must program its remote Setpoint Pointer in its loop parameter table location V+32, as shown below. The pointer must be the address of the V+05 location (control output) of the major loop. In Cascade Mode, the minor loop will ignore the its local SP register (V+02), and read the major loop's control output as its SP instead.



When using *Direct*SOFT's PID View to watch the SP value of the minor loop, *Direct*SOFT automatically reads the major loop's control output and displays it for the minor loop's SP. The minor loop's normal SP location, V+02, remains unchanged.

Now, we use the loop parameter arrangement above and draw its equivalent loop diagram, shown below.



Remember that a major loop goes to Manual Mode automatically if its minor loop is taken out of Cascade Mode.

Tuning Cascaded Loops

In tuning cascaded loops, you will need to de-couple the cascade relationship and tune the loops individually, using one of the loop tuning procedures previously covered.

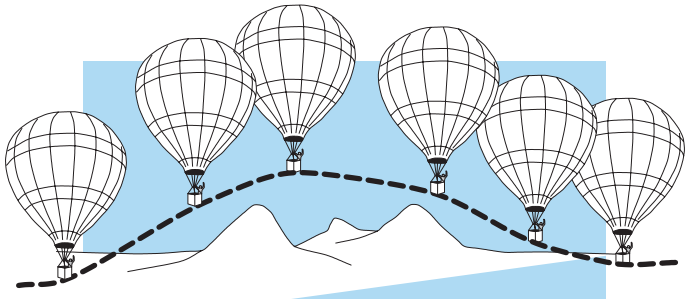
1. If you are not using auto tuning, then find the loop sample rate for the minor loop, using the method discussed earlier in this chapter. Then set the sample rate of the major loop slower than the minor loop by a factor of 10. Use this as a starting point.
2. Tune the minor loop first. Leave the major loop in Manual Mode, and you will need to generate SP changes for the minor loop manually as described in the loop tuning procedure.
3. Verify the minor loop gives a critically-damped response to a 10% SP change while in Auto Mode. Then we are finished tuning the minor loop.
4. In this step, you will need to get the minor loop in Cascade Mode, and then the Major loop in Auto Mode. We will be tuning the major loop with the minor loop treated as a series component its overall process. Therefore, do not go back and tune the minor loop again while tuning the major loop.
5. Tune the major loop, following the standard loop tuning procedure in this section. The response of the major loop PV is actually the overall response of the cascaded loops together.

Time-Proportioning Control

The PID loop controller in the DL205 CPU generates a smooth control output signal across a numerical range. The control output value is suitable to drive an analog output module, which connects to the process. In the process control field, this is called *continuous control*, because the output is on (at some level) continuously.

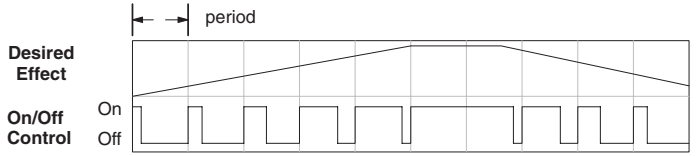
While continuous control can be smooth and robust, the cost of the loop components (such as actuator, heater amplifiers) can be expensive. A simpler form of control is called *time-proportioning control.* This method uses actuators which are either on or off (no in-between). Loop components for on/off-based control systems are lower cost than their continuous control counterparts.

In this section, we will show you how to convert the control output of a loop to time-proportioning control for the applications that need it. Let's take a moment to review how alternately turning a load on and off can control a process. The diagram below shows a hot-air balloon following a path across some mountains. The desired path is the *setpoint*. The balloon



pilot turns the burner on and off alternately, which is his *control output*. The large mass of air in the balloon effectively averages the effect of the burner, converting the bursts of heat into a continuous effect: slowly changing balloon temperature and ultimately the altitude, which is the *process variable*.

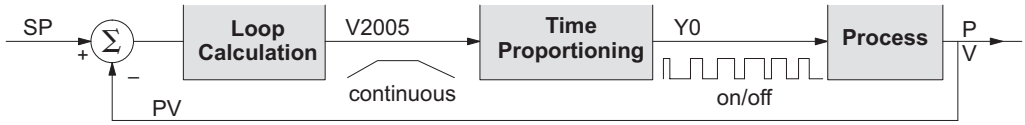
Time-proportioning control approximates continuous control by virtue of its duty-cycle – the ratio of ON time to OFF time. The following figure shows an example of how duty-cycle approximates a continuous level when it is averaged by a large process mass.



If we were to plot the on/off times of the burner in the hot-air balloon, we would probably see a very similar relationship to its effect on balloon temperature and altitude.

On/Off Control Program Example

The following ladder segment provides a time-proportioned on/off control output. It converts the continuous output in V2005 to on/off control using the output coil, Y0.

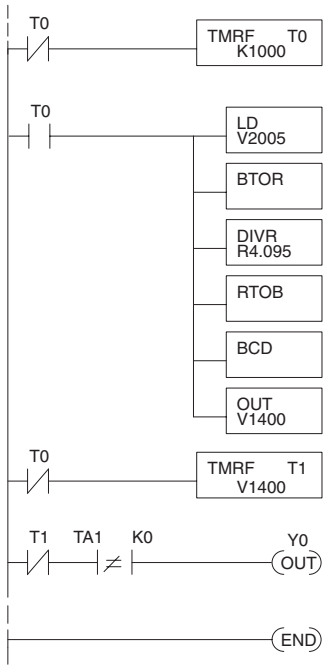


The example program uses two timers to generate On/Off control. It makes the following **assumptions**, which you can alter to fit your application:

- The loop table starts at V2000, so the control output is at V2005.
- The data format of the control output is 12-bit, unipolar (0 – FFF).
- The time base (one full cycle) for the On/Off waveform is 10 seconds. We use a fast timer (0.01 sec/tick), counting to 1000 ticks (10 seconds).
- The On/Off control output is Y0.

The time proportioning program must match the resolution of the output (1 part in 1000) to the resolution of the time base of T0 (also 1 part in 1000).

***NOTE**: Some processes change too fast for time proportioning control. Consider the speed of your process when you choose this control method. Use continuous control for processes that change too fast for time proportioning control. Also, consider using a solid state switch for a longer switch life instead of a relay.*



A fast timer (0.01 sec. timebase) establishes the primary time interval. The constant, K1000, sets the preset at 10 seconds (1,000 ticks). The N.C. enabling contact, T0, makes the timer self-resetting. T0 is on for one scan each 10 seconds, when it resets itself and T1.

At the end of the 10 second period, T0 turns on, and loads the control output value (binary) from the loop table V+05 location (V2005).

The BTOR instruction changes the number in the accumulator to a real number.

Dividing the control output by 4.095, converts the 0 – 4095 range to 0 – 1000, which "matches" the number of ticks in the 10 second timer range.

This instruction converts the real number back to binary. This step prepares the number for conversion to BCD. There is no real-to-BCD instruction.

Convert the number in the accumulator to BCD format. This satisfies the timer preset format requirement.

Output the result to V1400. In our example, this is the location of the timer preset for the second timer.

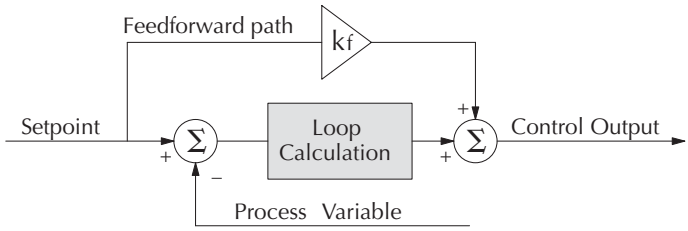
The second fast timer also counts in increments of .01 seconds, so its range is variable from 0 to a maximum of 1000 ticks, or 10 seconds. The timer T1 output, turns off the output coil, Y0, when the preset is reached.

The N.C. T1 contact, inverts the T1 timer output. The control output is on at the beginning of the 10-second time interval. Y0 turns off when T1 times out. The STRNE contact prevents Y0 from energizing during the one scan when T0 resets T1. Y0 is the actual control output.

END coil marks the end of the main program.

Feedforward Control

Feedforward control is an enhancement to standard closed-loop control. It is most useful for diminishing the effects of a *quantifiable and predictable* loop disturbance or sudden change in setpoint. Use of this feature is an option available to you on the DL205. However, it's best to implement and tune a loop without feedforward, and adding it only if better loop performance is still needed. The term *feedforward* refers to the control technique involved, shown in the diagram below. The incoming setpoint value is fed forward around the PID equation, and summed with the output.

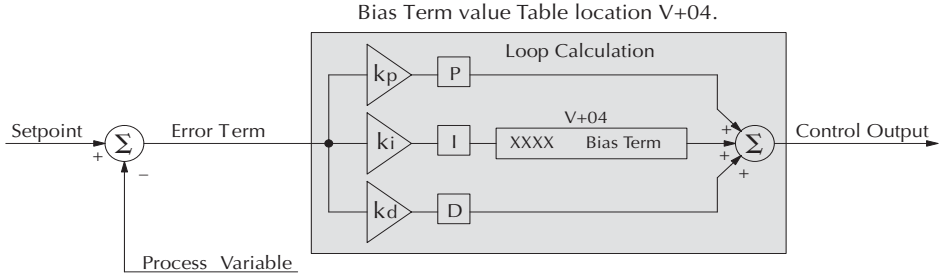


In the previous section on the bias term, we said that "the bias term value establishes a working region or operating point for the control output. *When the error fluctuates around its zero point, the output fluctuates around the bias value.*" Now, when there is a change in setpoint, an error is generated and the output must change to a new operating point. This also happens if a disturbance introduces a new offset in the loop. The loop does not really know its way to the new operating point. The integrator (bias) must increment/decrement until the error disappears, and then the bias has found the new operating point.

Suppose that we are able to know a sudden setpoint change is about to occur (common in some applications). We can avoid much of the resulting error in the first place if we can quickly change the output to the new operating point. If we know (from previous testing) what the operating point (bias value) will be after the setpoint change, we can artificially change the output directly (which is feedforward). The benefits of using feedforward are:

- The SP–PV error is reduced during predictable setpoint changes or loop offset disturbances.
- Proper use of feedforward will allow us to reduce the integrator gain. Reducing integrator gain gives us an even more stable control system.

Feedforward is very easy to use in the DL205 loop controller, as shown below. The bias term has been made available to the user in a special read/write location at PID Parameter Table location V+04.

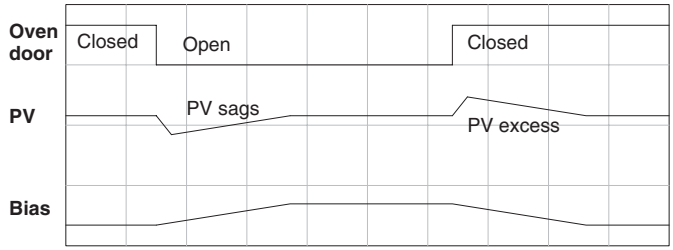


To change the bias (operating point), ladder logic only has to write the desired value to V+04. The PID loop calculation first reads the bias value from V+04 and modifies the value based on the current integrator calculation. Then it writes the result back to location V+04. This arrangement creates a sort of transparent bias term. All you have to do to implement feed forward control is write the correct value to the bias term at the right time (see the following example).

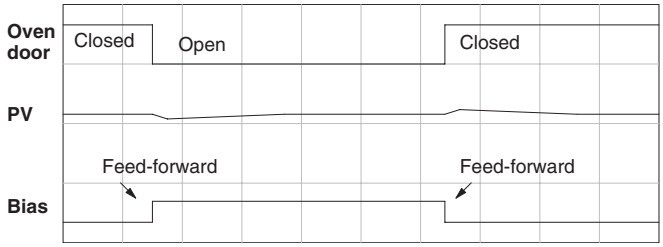
***NOTE**: When writing the bias term, one must be careful to design ladder logic to write the value only once, at the moment when the new bias operating point is to occur. If ladder logic writes the bias value on every scan, the loop's integrator is effectively disabled.*

Feedforward Example

How do we know when to write to the bias term, and what value to write? Suppose we have an oven temperature control loop, and we have already tuned the loop for optimal performance. Refer to the figure below. We notice that when the operator opens the oven door, the temperature sags a bit while the loop bias adjusts to the heat loss. Then, when the door closes, the temperature rises above the SP until the loop adjusts again. Feedforward control can help diminish this effect.



First, we record the amount of bias change the loop controller generates when the door opens or closes. Then, we write a ladder program to monitor the position of an oven door limit switch. When the door opens, our ladder program reads the current bias value from V+04, adds the desired change amount, and writes it back to V+04. When the door closes, we duplicate the procedure, but subtracting desired change amount instead. The following figure shows the results.



The step changes in the bias are the result of our two feed-forward writes to the bias term. We can see the PV variations are greatly reduced. The same technique may be applied for changes in setpoint.

PID Example Program

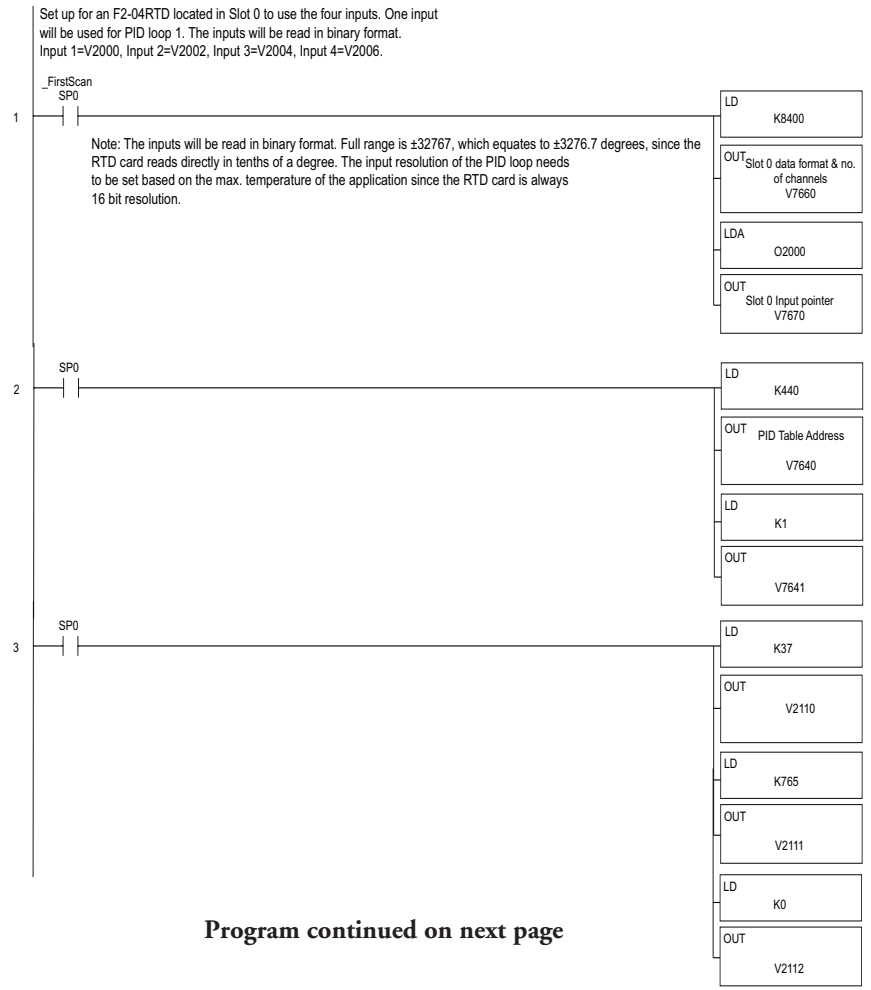
Program Setup for the PID Loop

After setting up the PID loop or loops, with *Direct*SOFT, you will need to edit your RLL program to include the rungs needed to set up the analog I/O module to be used by the PID loop(s).

The following example program shows how an RTD module, F2-04RTD, and an analog module, F2-02DA-2, are used and set up for a PID loop. This example assumes that the PID table for loop 1 has a beginning address of V2100.

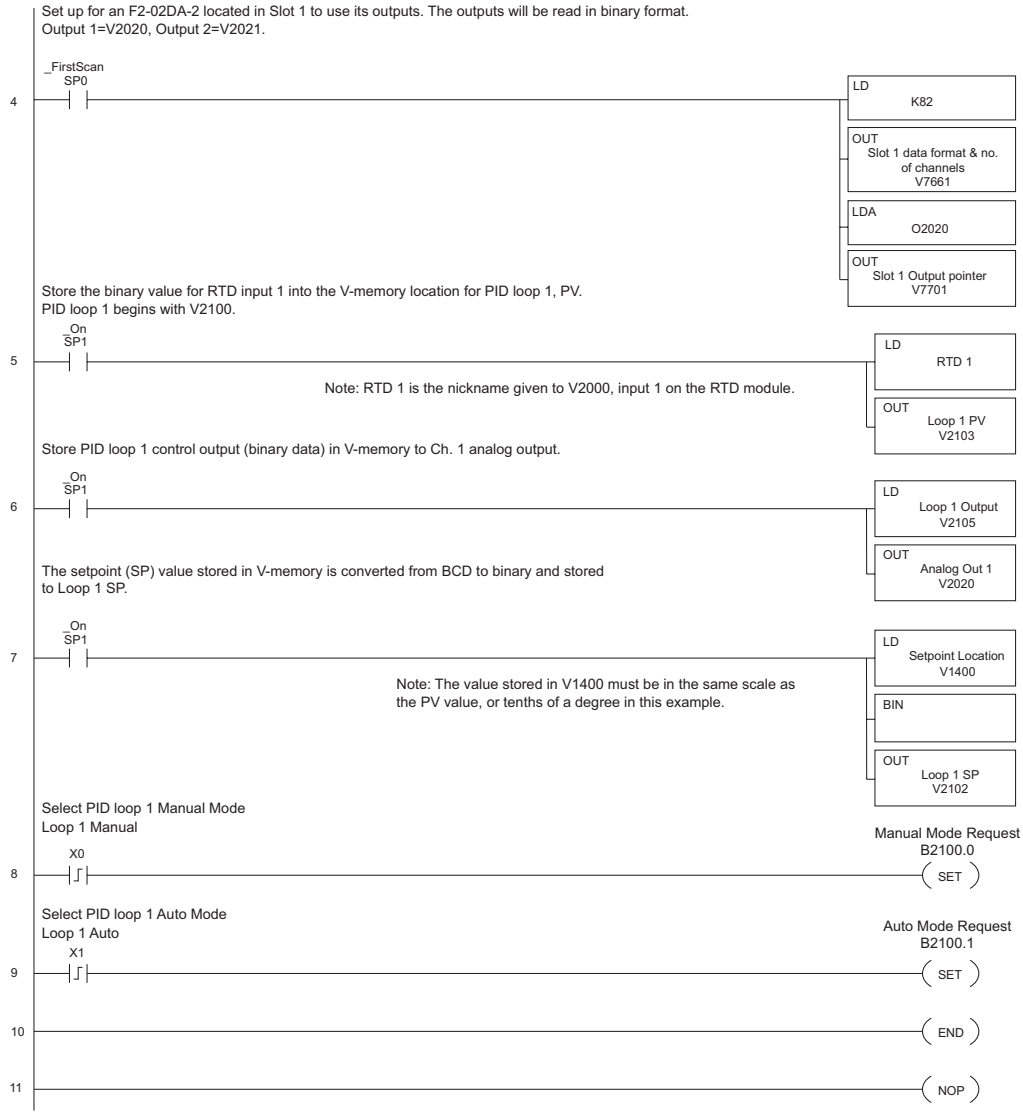
All of the analog I/O modules used with the DL205 are setup in a similar manner. Refer to the DL205 Analog Manual for the setup information for the particular module that you will be using.

***Direct*SOFT**



Program continued on next page

Example program continued



Note that the modules used in the PID loop example program were set up for binary format. They could have been set up for BCD format. In the latter case, the BCD data would have to be converted to binary format before being stored to the setpoint and process variable, and the control output would have to be converted from BCD to binary before being stored to the analog output.

By following the steps outlined in this chapter, you should be able to set up workable PID control loops. The ***Direct***SOFT Programming Software Manual provides more information for the use of PID View.

For a step-by-step tutorial, go to the Technical Support section located on our website, **www.automationdirect.com**. Once you are at the website, click on **Technical Support Home**. After this page opens, find and select **Guided Tutorials** located under the **Using Your Products** column. An **Animated Tutorial** page will open. Under **Available Tutorials**, find **PID Trainer** and select **View the PowerPoint slide show** and begin viewing the tutorial. The PowerPoint Viewer can be downloaded if your computer does not have PowerOoint installed.

Troubleshooting Tips

Q. The loop will not go into Automatic Mode.

A. Check the following for possible causes:

- A PV alarm exists, or a PV alarm programming error exists.
- The loop is the major loop of a cascaded pair, and the minor loop is not in Cascade Mode.

Q. The Control Output stays at zero constantly when the loop is in Automatic Mode.

A. Check the following for possible causes:

- The Control Output upper limit in loop table location V+31 is zero.
- The loop is driven into saturation, because the error never goes to zero value and changes (algebraic) sign.

Q. The Control Output value is not zero, but it is incorrect.

A. Check the following for possible causes:

- The gain values are entered improperly. Remember, gains are entered in the loop table in BCD, while the SP and PV are in binary. If you are using *Direct*SOFT, it displays the SP, PV, Bias and Control output in decimal (BCD), converting it to binary before updating the loop table.

Q. The Ramp/Soak Generator does not operate when I activate the Start bit.

A. Check the following for possible causes:

- The Ramp/Soak enable bit is off. Check the status of bit 11 of loop parameter table location V+00. It must be set =1.
- The hold bit or other bits in the Ramp/Soak control are on.
- The beginning SP value and the first ramp ending SP value are the same, so first ramp segment has no slope and consequently has no duration. The ramp/soak generator moves quickly to the soak segment, giving the illusion the first ramp is not working.
- The loop is in Cascade Mode, and is trying to get the SP remotely.
- The SP upper limit value in the loop table location V+27 is too low.
- Check your ladder program to verify it is not writing to the SP location (V+02 in the loop table). A quick way to do this is to temporarily place an end coil at the beginning of your program, then go to PLC Run Mode, and manually start the ramp/soak generator.

Q. The PV value in the table is constant, even though the analog module receives the PV signal.

A. Your ladder program must read the analog value from the module successfully and write it into the loop table V+03 location. Verify the analog module is generating the value, and the ladder is working.

Q. The Derivative gain doesn't seem to have any affect on the output.

A. The derivative limit is probably enabled (see section on derivative gain limiting).

Q. The loop Setpoint appears to be changing by itself.

A. Check the following for possible causes:

- The ramp/soak generator is enabled, and is generating setpoints.
- If this symptom occurs on loop Manual-to-Auto Mode changes, the loop is in Bumpless Transfer 1.
- Check your ladder program to verify it is not writing to the SP location (V+02 in the loop table). A quick way to do this is to temporarily place an end coil at the beginning of your program, then go to PLC Run Mode.

Q. The SP and PV values I enter with DirectSOFT work okay, but these values do not work properly when the ladder program writes the data.

A. The PID View in ***Direct***SOFT lets you enter SP, PV, and Bias values in decimal, and displays them in decimal for your convenience. For example, when the data format is 12 bit unipolar, the values range from 0 to 4095. However, the loop table actually requires these in hex, so ***Direct***SOFT converts them for you. The values in the table range from 0 to FFF, for 12-bit unipolar format.

Q. The loop seems unstable and impossible to tune, no matter what gains I use.

A. Check the following for possible causes:

- The loop sample time is set too long. Refer to the section near the front of this chapter on selecting the loop update time.
- The gains are too high. Start out by reducing the derivative gain to zero. Then reduce the integral gain, and the proportional gain if necessary.
- There is too much transfer lag in your process. This means the PV reacts sluggishly to control output changes. There may be too much "distance" between actuator and PV sensor, or the actuator may be weak in its ability to transfer energy into the process.
- There may be a process disturbance that is over-powering the loop. Make sure the PV is relatively steady when the SP is not changing.

Glossary of PID Loop Terminology

Automatic Mode: An operational mode of a loop, in which it makes PID calculations and updates the loop's control output.

Bias Freeze: A method of preserving the bias value (operating point) for a control output, by inhibiting the integrator when the output goes out of range. The benefit is a faster loop recovery.

Bias Term: In the position form of the PID equation, it is the sum of the integrator and the initial control output value.

Bumpless Transfer: A method of changing the operation mode of a loop while avoiding the usual sudden change in control output level. This consequence is avoided by artificially making the SP and PV equal, or the bias term and control output equal at the moment of mode change.

Cascaded Loops: A cascaded loop receives its setpoint from the output of another loop. Cascaded loops have a major/minor relationship, and work together to ultimately control one PV.

Cascade Mode: An operational mode of a loop, in which it receives its SP from another loop's output.

Continuous Control: Control of a process done by delivering a smooth (analog) signal as the control output.

Control Output: The numerical result of a PID equation which is sent by the loop with the intention of nulling out the current error.

Derivative Gain: A constant that determines the magnitude of the PID derivative term in response to the current error.

Direct-Acting Loop: A loop in which the PV increases in response to a control output increase. In other words, the process has a positive gain.

Error: The difference in value between the SP and PV, Error = SP – PV

Error Deadband: An optional feature which makes the loop insensitive to errors when they are small. You can specify the size of the deadband.

Error Squared: An optional feature which multiplies the error by itself, but retains the original algebraic sign. It reduces the effect of small errors, while magnifying the effect of large errors.

Feedforward: A method of optimizing the control response of a loop when a change in setpoint or disturbance offset is known and has a quantifiable effect on the bias term.

Integral Gain: A constant that determines the magnitude of the PID integral term in response to the current error.

Major Loop: In cascade control, it is the loop that generates a setpoint for the cascaded loop.

Manual Mode: An operational mode of a loop, in which the PID calculations are stopped. The operator must manually control the loop by writing to the control output value directly.

Minor Loop: In cascade control, the minor loop is the subordinate loop that receives its SP from the major loop.

On/Off Control: A simple method of controlling a process through on/off application of energy into the system. The mass of the process averages the on/off effect for a relatively smooth PV. A simple ladder program can convert the DL205's continuous loop output to on/off control.

PID Loop: A mathematical method of closed-loop control involving the sum of three terms based on proportional, integral, and derivative error values. The three terms have independent gain constants, allowing one to optimize (tune) the loop for a particular physical system.

Position Algorithm: The control output is calculated so it responds to the displacement (position) of the PV from the SP (error term)

Process: A manufacturing procedure which adds value to raw materials. Process control particularly refers to inducing chemical changes to the material in process.

Process Variable (PV): A quantitative measurement of a physical property of the material in process, which affects final product quality and is important to monitor and control.

Proportional Gain: A constant that determines the magnitude of the PID proportional term in response to the current error.

PV Absolute Alarm: A programmable alarm that compares the PV value to alarm threshold values.

PV Deviation Alarm: A programmable alarm that compares the difference between the SP and PV values to a deviation threshold value.

Ramp/Soak Profile: A set of SP values called a profile, which is generated in real time upon each loop calculation. The profile consists of a series of ramp and soak segment pairs, greatly simplifying the task of programming the PLC to generate such SP sequences.

Rate: Also called differentiator, the rate term responds to the changes in the error term.

Remote Setpoint: The location where a loop reads its setpoint when it is configured as the minor loop in a cascaded loop topology.

Reset: Also called integrator, the reset term adds each sampled error to the previous, maintaining a running total called the bias.

Reset Windup: A condition created when the loop is unable to find equilibrium, and the persistent error causes the integrator (reset) sum to grow excessively (windup). Reset windup causes an extra recovery delay when the original loop fault is remedied.

Reverse-Acting Loop: A loop in which the PV increases in response to a control output decrease. In other words, the process has a negative gain.

Sampling time: The time between PID calculations. The CPU method of process control is called a sampling controller, because it samples the SP and PV only periodically.

Setpoint (SP) The desired value for the process variable. The setpoint (SP) is the input command to the loop controller during closed loop operation.

Soak Deviation: The soak deviation is a measure of the difference between the SP and PV during a soak segment of the ramp/soak profile, when the ramp/soak generator is active.

Step Response: The behavior of the process variable in response to a step change in the SP (in closed loop operation), or a step change in the control output (in open loop operation).

Transfer: To change from one loop operational mode to another (between Manual, Auto, or Cascade). The word "transfer" probably refers to the transfer of control of the control output or the SP, depending on the particular mode change.

Velocity Algorithm: The control output is calculated to represent the rate of change (velocity) for the PV to become equal to the SP.

Bibliography

Fundamentals of Process Control Theory, Second Edition Author: Paul W. Murrill Publisher: Instrument Society of America ISBN 1–55617–297–4	Application Concepts of Process Control Author: Paul W. Murrill Publisher: Instrument Society of America ISBN 1–55617–080–7
PID Controllers: Theory, Design, and Tuning, 2nd Edition Author: K. Astrom and T Hagglund Publisher: Instrument Society of America ISBN 1–55617–516–7	Fundamentals of Temperature, Pressure, and Flow Measurements, Third edition Author: Robert P. Benedict Publisher: John Wiley and Sons ISBN 0–471–89383–8
Process / Industrial Instruments & Controls Handbook, Fourth Edition Author (Editor-in-Chief): Douglas M. Considine Publisher: McGraw-Hill, Inc ISBN 0-07-012445-0	pH Measurement and Control, Second Edition Author: Gregory K. McMillan Publisher: Instrument Society of America ISBN 1–55617–483–7
Programmable Controllers Concepts and Applications, First Edition Authors: C.T. Jones and L.A. Bryan Publisher: International Programmable Controls ISBN 0-915425-00-9	Fundamentals of Programmable Logic Controllers, Sensors, and Communications Author: Jon Stenerson Publisher: Prentice Hall ISBN 0-13-726860-2
Process Control, Third Edition Instrument Engineer's Handbook Author (Editor-in-Chief): Bela G. Liptak Publisher: Chilton ISBN 0–8019–8242–1	Process Measurement and Analysis, Third Edition Instrument Engineer's Handbook Author (Editor-in-Chief): Bela G. Liptak Publisher: Chilton ISBN 0–8019–8197–2

Maintenance and Troubleshooting

In This Chapter...

Hardware Maintenance

Standard Maintenance

The DL205 is a low maintenance system requiring only a few periodic checks to help reduce the risk of problems. Routine maintenance checks should be made regarding two key items.

- Air quality (cabinet temperature, airflow, etc), and
- CPU battery.

Air Quality Maintenance

The quality of the air your system is exposed to can affect system performance. If you have placed your system in an enclosure, check to see that the ambient temperature is not exceeding the operating specifications. If there are filters in the enclosure, clean or replace them as necessary to ensure adequate airflow. A good rule of thumb is to check your system environment every one to two months. Make sure the DL205 is operating within the system operating specifications.

Low Battery Indicator

The CPU has a battery LED that indicates the battery voltage is low. You should check this indicator periodically to determine if the battery needs replacing. You can also detect low battery voltage from within the CPU program. SP43 is a special relay that comes on when the battery needs to be replaced. If you are using a D2-240 CPU, you can also use a programming device or operator interface to determine the battery voltage. V7746 contains the battery voltage. For example, a value of 32 in V7746 would indicate a battery voltage of 3.2 V.

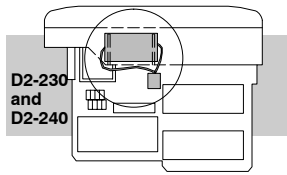
CPU Battery Replacement

The CPU battery is used to retain program V-memory and the system parameters. The life expectancy of this battery is five years.

NOTE: *Before installing or replacing your CPU battery, back up your V-memory and system parameters. You can do this by using DirectSOFT to save the program, V-memory, and system parameters to hard/ floppy disk on a personal computer.*

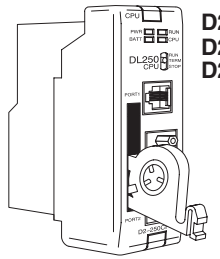
To install the D2–BAT CPU battery in D2-230 or D2-240 CPUs:

1. Gently push the battery connector onto the circuit board connector.
2. Push the battery into the retaining clip. Don't use excessive force. You may break the retaining clip.
3. Make a note of the date the battery was installed.



To install the D2–BAT–1 CPU battery in the D2-250–1, D2-260 and D2-262 CPUs: (#CR2354)

1. Press the retaining clip on the battery door down and swing the battery door open.
2. Remove the old battery and insert the new battery into the coin–type slot with the larger (+) side outwards.
3. Close the battery door, making sure that it locks securely in place.
4. Make a note of the date the battery was installed.



D2-250–1
D2-260
D2-262

WARNING: Do not attempt to recharge the battery or dispose of an old battery by fire. The battery may explode or release hazardous materials.

Diagnostics

Diagnostics

Your DL205 system performs many pre-defined diagnostic routines with every CPU scan. The diagnostics have been designed to detect various types of failures for the CPU and I/O modules. There are two primary error classes: fatal and non fatal.

Fatal Errors

Fatal errors are errors the CPU has detected that offer a risk of the system not functioning safely or properly. If the CPU is in Run Mode when the fatal error occurs, the CPU will switch to Program Mode. (Remember, in Program Mode all outputs are turned off.) If the fatal error is detected while the CPU is in Program Mode, the CPU will not enter Run Mode until the error has been corrected. Here are some examples of fatal errors.

- Base power supply failure
- Parity error or CPU malfunction
- I/O configuration errors
- Certain programming errors

Non-fatal Errors

Non-fatal errors are errors that are flagged by the CPU as requiring attention. They can neither cause the CPU to change from Run Mode to Program Mode, nor do they prevent the CPU from entering Run Mode. The application program can use the available special relays to detect if a non-fatal error has occurred. The application program can then be used to take the system to an orderly shutdown or to switch the CPU to Program Mode if necessary.

Some examples of non-fatal errors are:

- Back-up battery voltage low
- All I/O module errors
- Certain programming errors

Finding Diagnostic Information

Diagnostic information can be found in several places with varying levels of message detail.

- The CPU automatically logs error codes and any FAULT messages into two separate tables which can be viewed with the Handheld Programmer or DirectSOFT.
- The Handheld Programmer displays error numbers and short descriptions of the error.
- DirectSOFT provides the error number and an error message.
- Appendix B in this manual has a complete list of error messages sorted by error number.

Many of these messages point to supplemental memory locations which can be referenced for additional related information. These memory references are in the form of V-memory and SPs (special relays).

The following two tables name the specific memory locations that correspond to certain types of error messages. The special relay table also includes status indicators which can be used in programming. For a more detailed description of each of these special relays, refer to Appendix D.

V-memory Locations Corresponding to Error Codes

Error Class	Error Category	Diagnostic V-memory
Battery Voltage (D2-240 only)	Shows battery voltage to tenths (32 is 3.2V)	V7746
User-Defined	Error code used with FAULT instruction	V7751
I/O Configuration	Correct module ID code	V7752
	Incorrect module ID code	V7753
	Base and Slot number where error occurs	V7754
System Error	Fatal Error code	V7755
	Major Error code	V7756
	Minor Error code	V7757
Module Diagnostic	Base and slot number where error occurs	V7760
	Always holds a “0”	V7761
	Error code	V7762
Grammatical	Address where syntax error occurs	V7763
	Error code found during syntax check	V7764
CPU Scan	Number of scans since last Program to Run Mode transition	V7765
	Current scan time (ms)	V7775
	Minimum scan time (ms)	V7776
	Maximum scan time (ms)	V7777

Special Relays (SP) Corresponding to Error Codes

Startup and Real-time Relays	
SP0	On first scan only
SP1	Always ON
SP2	Always OFF
SP3	1 minute clock
SP4	1 second clock
SP5	100 millisecond clock
SP6	50 millisecond clock
SP7	On alternate scans
CPU Status Relays	
SP11	Forced Run mode (D2-240/D2-250-1/D2-260/D2-262)
SP12	Terminal Run mode
SP13	Test Run mode (D2-240/D2-250-1/D2-260/D2-262)
SP15	Test program mode (D2-240/D2-250-1/D2-260/D2-262)
SP16	Terminal Program mode
SP20	STOP instruction was executed
SP22	Interrupt enabled
System Monitoring Relays	
SP40	Critical error
SP41	Non-critical error
SP43	Battery low
SP44	Program memory error
SP45	I/O error
SP46	Communications error
SP47	I/O configuration error
SP50	Fault instruction was executed
SP51	Watchdog timeout
SP52	Syntax error
SP53	Cannot solve the logic
SP54	Intelligent module communication error

Accumulator Status Relays	
SP60	Acc. is less than value
SP61	Acc. is equal to value
SP62	Acc. is greater than value
SP63	Acc. result is zero
SP64	Half borrow occurred
SP65	Borrow occurred
SP66	Half carry occurred
SP67	Carry occurred
SP70	Result is negative (sign)
SP71	Pointer reference error
SP73	Overflow
SP75	Data is not in BCD
SP76	Load zero
Communication Monitoring Relays	
SP116 (D2-230/D2-240)	CPU is communicating with another device
SP116 (D2-250-1/ D2-260/D2-262)	Port 2 is communicating with another device
SP117	Communication error on Port 2 (D2-250-1/D2-260/D2-262 only)
SP120	Module busy, Slot 0
SP121	Communication error Slot 0
SP122	Module busy, Slot 1
SP123	Communication error Slot 1
SP124	Module busy, Slot 2
SP125	Communication error Slot 2
SP126	Module busy, Slot 3
SP127	Communication error Slot 3
SP130	Module busy, Slot 4
SP131	Communication error Slot 4
SP132	Module busy, Slot 5
SP133	Communication error Slot 5
SP134	Module busy, Slot 6
SP135	Communication error Slot 6
SP136	Module busy, Slot 7
SP137	Communication error Slot 7

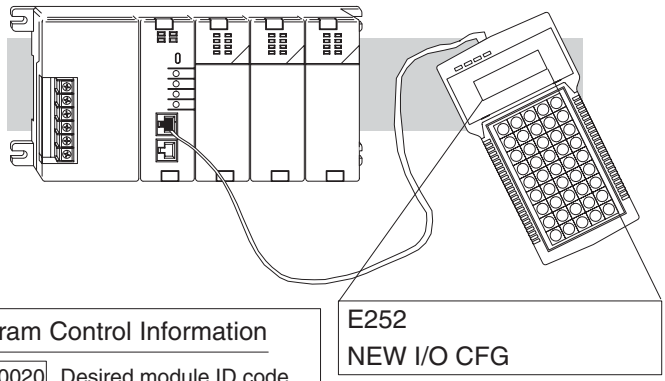
I/O Module Codes

Each system component has a code identifier. This code identifier is used in some of the error messages related to the I/O modules. The following table shows these codes.

Code (Hex)	Component Type
04	CPU
03	I/O Base
20	8 pt. Output
21	8 pt. Input
24	4 input/output combination
28	12 pt. Output 16 pt. Output
3F	32 pt. Input
30	32 pt. Output
52	H2-ERM(100)
51	H2-CTRIO(2)

Code (Hex)	Component Type
36	Analog Input
2B	16 pt. Input
37	Analog Output
3D	Analog I/O Combo
4A	Counter Interface
7F	Abnormal
FF	No module detected
EE	D2-DCM H2-ECOM F2-CP128
BE	D2-RMSM

The following diagram shows an example of how the I/O module codes are used:



Program Control Information

V7752	0020	Desired module ID code
V7753	0026	Current module ID code
V7754	0002	Location of conflict
V7755	0252	Fatal error code
SP47	⊣⊢	I/O Configuration Error

Error Message Tables

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The D2-240, D2-250-1, D2-260 and D2-262 CPUs will automatically log any system error codes and any custom messages you have created in your application program with the FAULT instructions. The CPU logs the error code, the date, and the time the error occurred. Two separate tables store this information.

- Error Code Table – the system logs up to 32 errors in the table. When an error occurs, the errors already in the table are pushed down and the most recent error is loaded into the top position. If the table is full when an error occurs, the oldest error is pushed (erased) from the table.
- Message Table – the system logs up to 16 messages in this table. When a message is triggered, the messages already stored in the table are pushed down and the most recent message is loaded into the top position. If the table is full when an error occurs, the oldest message is pushed (erased) from the table.

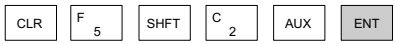
Date	Time	Message
2008-05-26	08:41:51:11	*Conveyor-2 stopped
2008-04-30	17:01:11:56	*Conveyor-1 stopped
2008-04-30	17:01:11:12	*Limit SW1 failed
2008-04-28	03:25:14:31	*Saw Jam Detect

The following table shows an example of an error table for messages.

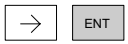
You can access the error code table and the message table through *Direct*SOFT's PLC Diagnostic sub-menus or from the Handheld Programmer. Details on how to access these logs are provided in the *Direct*SOFT manual.

The following examples show you how to use the Handheld Programmer and AUX Function 5C to show the error codes. The most recent error or message is always displayed. You can use the PREV and NXT keys to scroll through the messages.

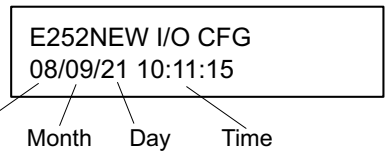
Use AUX 5C to view the tables



Use the arrow key to select Errors or Messages



Example of an error display



System Error Codes

☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

The System error log contains 32 of the most recent errors that have been detected. The errors that are trapped in the error log are a subset of all the error messages which the DL205 systems generate. These errors can be generated by the CPU or by the Handheld Programmer, depending on the actual error. Appendix B provides more complete descriptions of the error codes.

The errors can be detected at various times. However, most of them are detected at power-up, on entry to Run Mode, or when a Handheld Programmer key sequence results in an error or an illegal request.

Error Code	Description
E003	Software time-out
E004	Invalid instruction (RAM parity error in the CPU)
E041	CPU battery low
E043	Memory cartridge battery low
E099	Program memory exceeded
E101	CPU memory cartridge missing
E104	Write fail
E151	Invalid command
E155	RAM failure
E201	Terminal block missing
E202	Missing I/O module
E203	Blown fuse
E206	User 24V power supply failure
E210	Power fault
E250	Communication failure in the I/O chain
E251	I/O parity error
E252	New I/O configuration
E262	I/O out of range
E312	Communications error 2
E313	Communications error 3
E316	Communications error 6
E320	Time out
E321	Communications error
E499	Invalid Text entry for Print Instruction
E501	Bad entry
E502	Bad address
E503	Bad command
E504	Bad reference / value
E505	Invalid instruction

Error Code	Description
E506	Invalid operation
E520	Bad operation - CPU in Run
E521	Bad operation - CPU in Test Run
E523	Bad operation - CPU in Test Program
E524	Bad operation - CPU in Program
E525	Mode Switch not in TERM
E526	Unit is offline
E527	Unit is online
E528	CPU mode
E540	CPU locked
E541	Wrong password
E542	Password reset
E601	Memory full
E602	Instruction missing
E604	Reference missing
E610	Bad I/O type
E611	Bad Communications ID
E620	Out of memory
E621	EEPROM Memory not blank
E622	No Handheld Programmer EEPROM
E624	V-memory only
E625	Program only
E627	Bad write operation
E628	Memory type error (should be EEPROM)
E640	Mis-compare
E650	Handheld Programmer system error
E651	Handheld Programmer ROM error
E652	Handheld Programmer RAM error

Program Error Codes

The following list shows the errors that can occur when there are problems with the program. These errors will be detected when you try to place the CPU into Run Mode or when you use AUX 21 – Check Program. The CPU will also turn on SP52 and store the error code in V7755. Appendix B provides more complete descriptions of the error codes.

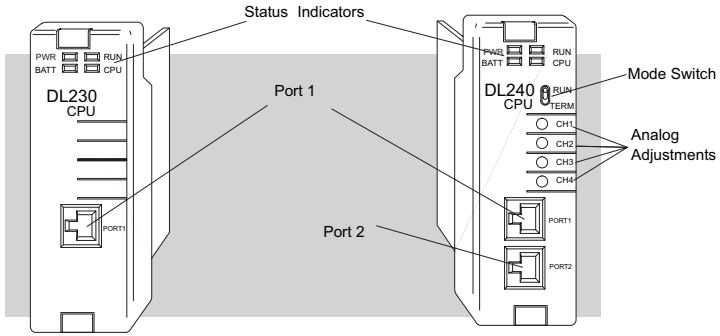
Error Code	Description
E4**	No Program in CPU
E401	Missing END statement
E402	Missing LBL
E403	Missing RET
E404	Missing FOR
E405	Missing NEXT
E406	Missing IRT
E412	SBR/LBL >64
E413	FOR/NEXT >64
E421	Duplicate stage reference
E422	Duplicate SBR/LBL reference
E423	Nested loops
E431	Invalid ISG/SG address
E432	Invalid jump (GOTO) address
E433	Invalid SBR address
E434	Invalid RTC address
E435	Invalid RT address
E436	Invalid INT address
E437	Invalid IRTC address
E438	Invalid IRT address
E440	Invalid Data address
E441	ACON/NCON
E451	Bad MLS/MLR
E452	X input used as output coil
E453	Missing T/C
E454	Bad TMRA
E455	Bad CNT
E456	Bad SR

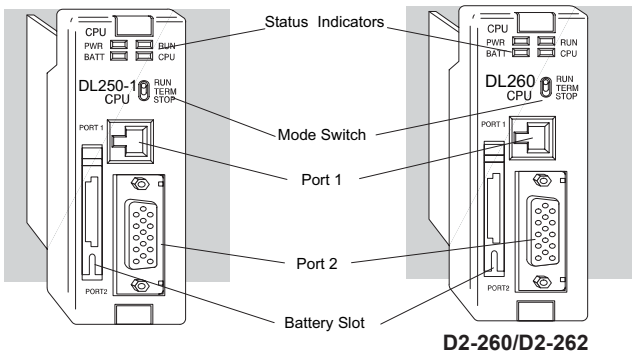
Error Code	Description
E461	Stack Overflow
E462	Stack Underflow
E463	Logic Error
E464	Missing Circuit
E471	Duplicate coil reference
E472	Duplicate TMR reference
E473	Duplicate CNT reference
E480	CV position error
E481	CV not connected
E482	CV exceeded
E483	CVJMP placement error
E484	No CV
E485	No CVJMP
E486	BCALL placement error
E487	No Block defined
E488	Block position error
E489	Block CR identifier error
E490	No Block stage
E491	ISG position error
E492	BEND position error
E493	BEND I error
E494	No BEND

CPU Error Indicators

The DL205 CPUs have indicators on the front to help you diagnose problems with the system. The table below gives a quick reference of potential problems associated with each status indicator. Following the table will be a detailed analysis of each of these indicator problems.

Indicator Status	Potential Problems
PWR (off)	1. System voltage incorrect
	2. Power supply/CPU is faulty
	3. Other component such an I/O module has power supply shorted
	4. Power budget exceeded for the base being used
RUN (will not come on)	1. CPU programming error
	2. Switch in TERM position
	3. Switch in STOP position (D2-250-1, D2-260 and D2-262 only)
RUN (flashing)	Firmware upgrade mode
CPU (on)	1. Electrical noise interference
	2. CPU defective
BATT (on)	1. CPU battery low
	2. CPU battery missing, or disconnected





PWR Indicator

There are four general reasons for the CPU power status LED (PWR) to be OFF:

- Power to the base is incorrect or is not applied.
- Base power supply is faulty.
- Other component(s) have the power supply shut down.
- Power budget for the base has been exceeded.

Incorrect Base Power

If the voltage to the power supply is not correct, the CPU and/or base may not operate properly or may not operate at all. Use the following guidelines to correct the problem.

WARNING: To minimize the risk of electrical shock, always disconnect the system power before inspecting the physical wiring.

1. First, disconnect the system power and check all incoming wiring for loose connections.
2. If you are using a separate termination panel, check those connections to make sure the wiring is connected to the proper location.
3. If the connections are acceptable, reconnect the system power and measure the voltage at the base terminal strip to ensure it is within specification. If the voltage is not correct, shut down the system and correct the problem.
4. If all wiring is connected correctly and the incoming power is within the specifications required, the base power supply should be returned for repair.

Faulty CPU

There is not a good check to test for a faulty CPU other than substituting a known good one to see if this corrects the problem. If you have experienced major power surges, it is possible the CPU and power supply have been damaged. If you suspect this is the cause of the power supply damage, a line conditioner that removes damaging voltage spikes should be used in the future.

Device or Module causing the Power Supply to Shutdown

It is possible a faulty module or external device using the system 5V can shut down the power supply. This 5V can be coming from the base or from the CPU communication ports.

To test for a device causing this problem:

1. Turn off power to the CPU.
2. Disconnect all external devices (i.e., communication cables) from the CPU.
3. Reapply power to the system.

If the power supply operates normally, you may have either a shorted device or a shorted cable. If the power supply does not operate normally, then test for a module causing the problem by following the steps below:

If the PWR LED operates normally, the problem could be in one of the modules. To isolate which module is causing the problem, disconnect the system power and remove one module at a time until the PWR LED operates normally. Follow the procedure below:

- Turn off power to the base.
- Remove a module from the base.
- Reapply power to the base.

Bent base connector pins on the module can cause this problem. Check to see the connector is not the problem.

Power Budget Exceeded

If the machine had been operating correctly for a considerable amount of time prior to the indicator going off, the power budget is not likely to be the problem. Power budgeting problems usually occur during system startup when the PLC is under operation and the inputs/outputs are requiring more current than the base power supply can provide.

WARNING: The PLC may reset if the power budget is exceeded. If there is any doubt about the system power budget, please check it at this time. Exceeding the power budget can cause unpredictable results which can cause damage and injury. Verify the modules in the base operate within the power budget for the chosen base. You can find these tables in Chapter 4, Bases and I/O Configuration.

Run Indicator

If the CPU will not enter the Run mode (the RUN indicator is off), the problem is usually in the application program, unless the CPU has a fatal error. If a fatal error has occurred, the CPU LED should be on. (You can use a programming device to determine the cause of the error.) If the RUN light is flashing, the PLC is in firmware upgrade mode.

If you are using a D2-240, D2-250–1, D2-260 or D2-262 and you are trying to change the modes with a programming device, make sure the mode switch is in the TERM position.

Both of the programming devices, Handheld Programmer and *Direct*SOFT, will return an error message describing the problem. Depending on the error, there may also be an AUX function you can use to help diagnose the problem. The most common programming error is "Missing END Statement." All application programs require an END statement for proper termination. A complete list of error codes can be found in Appendix B.

CPU Indicator

If the CPU indicator is on, a fatal error has occurred in the CPU. Generally, this is not a programming problem but an actual hardware failure. You can power cycle the system to clear the error. If the error clears, you should monitor the system and determine what caused the problem. You will find this problem is sometimes caused by high frequency electrical noise introduced into the CPU from an outside source. Check your system grounding and install electrical noise filters if the grounding is suspected. If power cycling the system does not reset the error, or if the problem returns, you should replace the CPU.

BATT Indicator

If the BATT indicator is on, the CPU battery is either disconnected or needs replacing. The battery voltage is continuously monitored while the system voltage is being supplied.

Communications Problems

If you cannot establish communications with the CPU, check these items.

- The cable is disconnected.
- The cable has a broken wire or has been wired incorrectly.
- The cable is improperly terminated or grounded.
- The device connected is not operating at the correct baud rate (9600 baud for the top port. Use AUX 56 to select the baud rate for the bottom port on a D2-240, D2-250–1, D2-260 or D2-262).
- The device connected to the port is sending data incorrectly.
- A grounding difference exists between the two devices.
- Electrical noise is causing intermittent errors.
- The CPU has a bad communication port and the CPU should be replaced.

If an error occurs, the indicator will come on and stay on until a successful communication has been completed.

I/O Module Troubleshooting

Things to Check

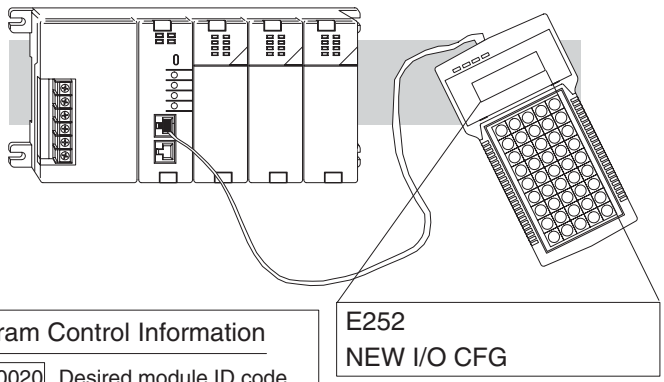
If you suspect an I/O error, there are several things that could be causing the problem.

- A blown fuse.
- A loose terminal block.
- The 24VDC supply has failed.
- The module has failed.
- The I/O configuration check detects a change in the I/O configuration.

I/O Diagnostics

If the modules are not providing any clues to the problem, run AUX 42 from the handheld programmer or I/O diagnostics in ***Direct***SOFT. Both options will provide the base number, the slot number and the problem with the module. Once the problem is corrected, the indicators will reset.

An I/O error will not cause the CPU to switch from the run to program mode; however there are special relays (SPs) available in the CPU which will allow this error to be read in ladder logic. The application program can then take the required action, such as entering the program mode or initiating an orderly shutdown. The figure below shows an example of the failure indicators.



Program Control Information

V7752	0020	Desired module ID code
V7753	0021	Current module ID code
V7754	0002	Location of conflict
V7755	0252	Fatal error code
SP47	⊣⊢	I/O Configuration Error

Some Quick Steps

When troubleshooting the DL205 series I/O modules, you should be aware of a few facts you that may assist you in quickly correcting an I/O problem:

- The output modules cannot detect shorted or open output points. If you suspect one or more points on an output module to be faulty, you should measure the voltage drop from the common to the suspect point. Remember, when using a Digital Volt Meter, leakage current from an output device, such as a triac or a transistor, must be considered. A point which is off may appear to be on if no load is connected to the point.
- The I/O point status indicators on the modules are logic side indicators. This means the LED which indicates the on or off status reflects the status of the point with respect to the CPU. On an output module, the status indicators could be operating normally, while the actual output device (transistor, triac etc) could be damaged. With an input module, if the indicator LED is on, the input circuitry should be operating properly. To verify proper functionality, check to see that the LED goes off when the input signal is removed.
- Leakage current can be a problem when connecting field devices to I/O modules. False input signals can be generated when the leakage current of an output device is great enough to turn on the connected input device. To correct this, install a resistor in parallel with the input or output of the circuit. The value of this resistor will depend on the amount of leakage current and the voltage applied but usually a 10kΩ to 20kΩ resistor will work. Ensure the wattage rating of the resistor is correct for your application.
- The easiest method to determine if a module has failed is to replace it, if you have a spare. However, if you suspect another device to have caused the failure in the module, that device may cause the same failure in the replacement module as well. As a point of caution, you may want to check devices or power supplies connected to the failed module before replacing it with a spare module.

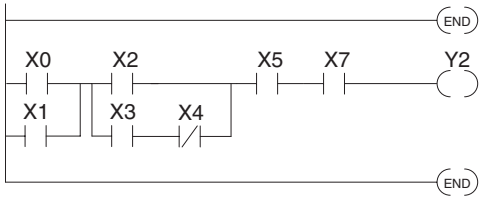
Testing Output Points

Output points can be set on or off in the DL205 series CPUs. In the D2-240 and D2-250-1, you can use AUX 59, Bit Override, to force a point even while the program is running. However, this is not a recommended method to test the output points. If you want to do an I/O check independent of the application program for either the D2-230, D2-240, D2-250-1, D2-260 or D2-262, follow the procedure below:

Step	Action
1	Use a Handheld Programmer or ***Direct***SOFT to communicate online to the PLC.
2	Change to Program Mode.
3	Go to address 0.
4	Insert an "END" statement at address 0. (This will cause program execution to occur only at address 0 and prevent the application program from turning the I/O points on or off.
5	Change to Run Mode.
6	Use the programming device to set (turn) on or off the points you wish to test.
7	When you finish testing I/O points, delete the "END" statement at address 0.

WARNING: Depending on your application, forcing I/O points may cause unpredictable machine operation that can result in a risk of personal injury or equipment damage. Make sure you have taken all appropriate safety precautions prior to testing any I/O points.

Handheld Programmer Keystrokes Used to Test an Output Point



Insert an END statement at the beginning of the program. This disables the remainder of the program.

From a clear display, use the following keystrokes

STAT ENT

```
16P STATUS
BIT REF   X
```

Use the PREV or NEXT keys to select the Y data type

NEXT A 0 ENT

```
 Y  10   Y   0
□□□□□□□□□□□□□□□□
```

Use arrow keys to select point, then use ON and OFF to change the status

← ← SHFT ON INS

Y2 is now on

```
 Y  10   Y   0
□□□□□□□□□□□□□■□□
```

Noise Troubleshooting

Electrical Noise Problems

Noise is one of the most difficult problems to diagnose. Electrical noise can enter a system in many different ways and falls into one of two categories: conducted or radiated. It may be difficult to determine how the noise is entering the system, but the corrective actions for either type of noise problem are similar.

- Conducted noise is when the electrical interference is introduced into the system by way of an attached wire, panel connection, etc. It may enter through an I/O module, a power supply connection, the communication ground connection, or the chassis ground connection.
- Radiated noise is when the electrical interference is introduced into the system without a direct electrical connection, much in the same manner as radio waves.

Reducing Electrical Noise

While electrical noise cannot be eliminated, it can be reduced to a level that will not affect the system.

- Most noise problems result from improper grounding of the system. A good earth ground can be the single most effective way to correct noise problems. If a ground is not available, install a ground rod as close to the system as possible. Ensure all ground wires are single-point grounds and are not daisy chained from one device to another. Ground metal enclosures around the system. A loose wire is no more than a large antenna waiting to introduce noise into the system; therefore, you should tighten all connections in your system. Loose ground wires are more susceptible to noise than the other wires in your system. Review Chapter 2, Installation, Wiring, and Specifications, if you have questions regarding how to ground your system.
- Electrical noise can enter the system through the power source for the CPU and I/O. Installing an isolation transformer for all AC sources can correct this problem. DC power sources should be well grounded, good quality power supplies. Switching DC power supplies commonly generate more noise than linear supplies.
- Separate input wiring from output wiring. Never run I/O wiring close to high voltage wiring.

Machine Startup and Program Troubleshooting

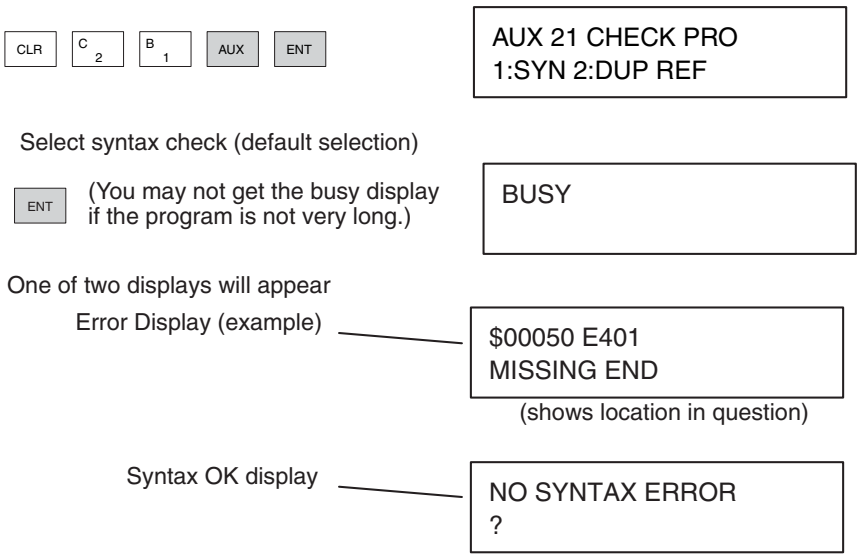
The DL205 CPUs provide several features to help you debug your program before and during machine startup. This section discusses the following topics which can be very helpful.

- Program Syntax Check
- Duplicate Reference Check
- Test Modes
- Special Instructions
- Run Time Edits
- Forcing I/O Points

Program Syntax Check

Even though the Handheld Programmer and *Direct*SOFT provide error checking during program entry, you may want to check a modified program. Both programming devices offer a way to check the program syntax. For example, you can use AUX 21, CHECK PROGRAM to check the program syntax from a Handheld Programmer, or you can use the PLC Diagnostics menu option within *Direct*SOFT. This check will find a wide variety of programming errors. The following example shows how to use the syntax check with a Handheld Programmer.

Use AUX 21 to perform syntax check

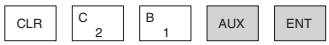


See the Error Codes Section for a complete listing of programming error codes. If you get an error, press CLR and the Handheld Programmer will display the instruction where the error occurred. Correct the problem and continue running the Syntax check until the NO SYNTAX ERROR message appears.

Duplicate Reference Check

You can also check for multiple uses of the same output coil. Both programming devices offer a way to check for this condition. For example, you can AUX 21, CHECK PROGRAM to check for duplicate references from a Handheld Programmer, or you can use the PLC Diagnostics menu option within *Direct*SOFT. The following example shows how to perform the duplicate reference check with a Handheld Programmer.

Use AUX 21 to perform syntax check




```
AUX 21 CHECK PRO
1:SYN 2:DUP REF
```

Select duplicate reference check

(You may not get the busy display if the program is not very long.)

```
BUSY
```

One of two displays will appear

Error Display (example)

```
$00024 E471
DUP COIL REF
```

(shows location in question)

Syntax OK display

```
NO DUP REFS
?
```

If you get an error, press CLR and the Handheld Programmer will display the instruction where the error occurred. Correct the problem and continue running the Duplicate Reference check until no duplicate references are found.

NOTE: *You can use the same coil in more than one location, especially in programs using the Stage instructions and/or the OROUT instructions. The Duplicate Reference check will find these outputs even though they may be used in an acceptable fashion.*

TEST-PGM and TEST-RUN Modes

Test Mode allows the CPU to start in TEST-PGM mode, enter TEST-RUN mode, run a fixed number of scans, and return to TEST-PGM mode. You can select from 1 to 65,525 scans. Test Mode also allows you to maintain output status while you switch between Test-Program and Test-Run Modes. You can select Test Modes from either the Handheld Programmer (by using the MODE key) or from *Direct*SOFT via a PLC Modes menu option.

The primary benefit of using the TEST mode is to maintain certain outputs and other parameters when the CPU transitions back to Test-Program mode. For example, you can use AUX 58 from the DL205 Handheld Programmer to configure the individual outputs, CRs, etc., to hold their output state. Also, the CPU will maintain timer and counter current values when it switches to TEST-PGM mode.

***NOTE**: You can only use **Direct**SOFT to specify the number of scans. This feature is not supported on the Handheld Programmer. However, you can use the Handheld Programmer to switch between Test Program and Test Run Modes.*

With the Handheld Programmer, the actual mode entered when you first select Test Mode depends on the mode of operation at the time you make the request. If the CPU is in Run Mode, then TEST-RUN is available. If the mode is Program, then TEST-PGM is available. Once you've selected TEST Mode, you can easily switch between TEST-RUN and TEST-PGM. *Direct*SOFT provides more flexibility in selecting the various modes with different menu options. The following example shows how you can use the Handheld Programmer to select the Test Modes.

Use the MODE key to select TEST Modes (example assumes Run Mode)

MODE NEXT ENT

```
*MODE CHANGE*
GO TO T–RUN MODE
```

Press ENT to confirm TEST-RUN Mode

ENT (Note, the TEST LED on the DL205 Handheld indicates the CPU is in TEST Mode.)

```
*MODE CHANGE*
CPU T–RUN
```

You can return to Run Mode, enter Program Mode, or enter TEST-PGM Mode by using the Mode Key

CLR MODE NEXT NEXT ENT

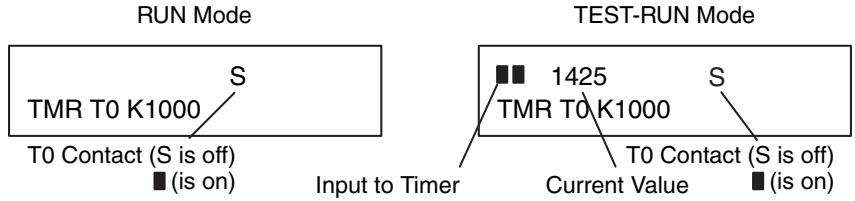
```
*MODE CHANGE*
GO TO T–PGM MODE
```

Press ENT to confirm TEST-PGM Mode

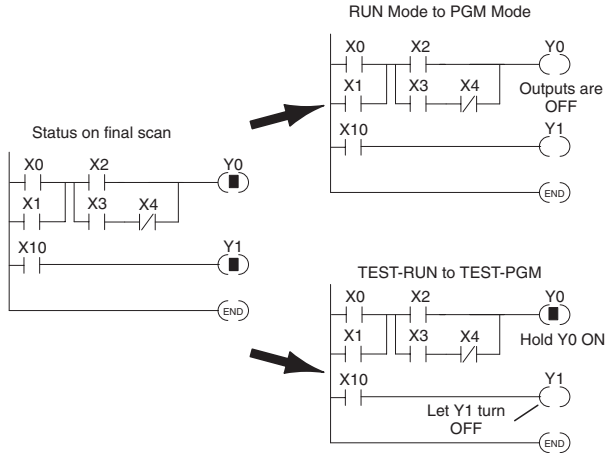
ENT (Note, the TEST LED on the DL205 Handheld indicates the CPU is in TEST Mode.)

```
*MODE CHANGE*
CPU T–PGM
```

Test Displays: With the Handheld Programmer you also have a more detailed display when you use TEST Mode. For some instructions, the TEST-RUN mode display is more detailed than the status displays shown in RUN mode. The following diagram shows an example of a Timer instruction display during TEST-RUN mode.



Holding Output States: The ability to hold output states is very useful because it allows you to maintain key system I/O points. In some cases, you may need to modify the program, but you do not want certain operations to stop. In normal Run Mode, the outputs are turned off when you return to Program Mode. In TEST-RUN mode, you can set each individual output to either turn off, or to hold its last output state on the transition to TEST-PGM mode. You can use AUX 58 on the Handheld Programmer to select the action for each individual output. This feature is also available via a menu option within ***Direct***SOFT. The following diagram shows the differences between RUN and TEST-RUN modes.



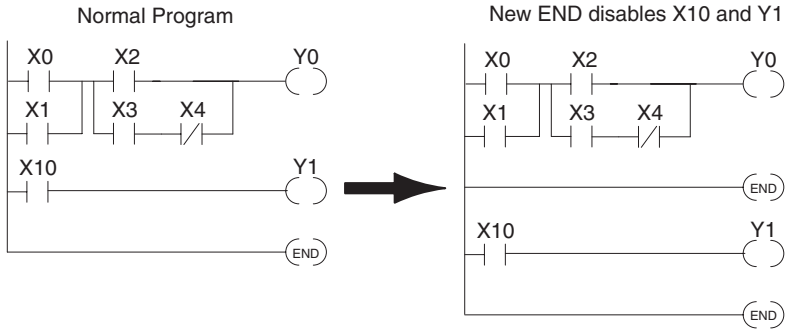
Before you decide that Test Mode is the perfect choice, remember that the DL205 CPUs also allow you to edit the program during Run Mode. The primary difference between the Test Modes and the Run Time Edit feature is you do not have to configure each individual I/O point to hold the output status. When you use Run Time Edits, the CPU automatically maintains all outputs in their current states while the program is being updated.

Special Instructions

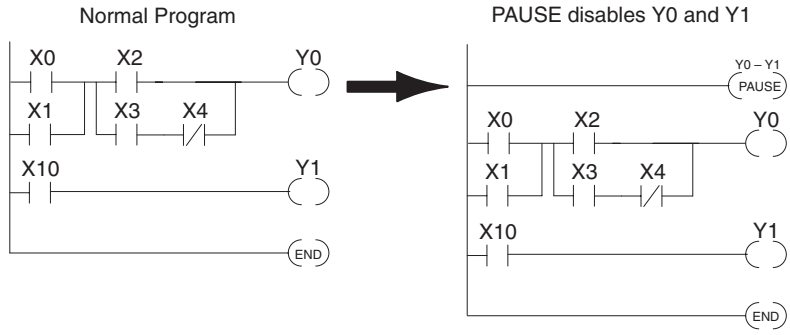
Several instructions can be used to help you debug your program during machine start-up operations.

- END
- PAUSE
- STOP

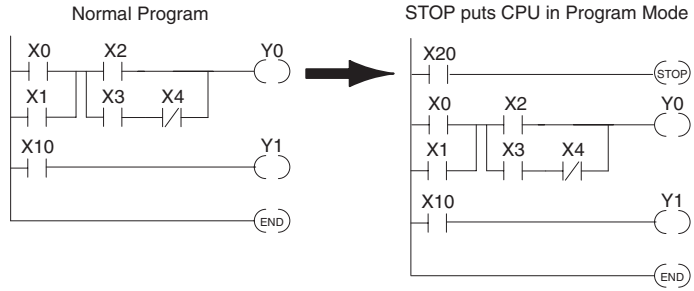
END Instruction: If you need a way to quickly disable part of the program, insert an END statement prior to the portion that should be disabled. When the CPU encounters the END statement, it assumes it is the end of the program. The following diagram shows an example.



PAUSE Instruction: This instruction provides a quick way to allow the inputs (or other logic) to operate while disabling selected outputs. The output image register is still updated, but the output status is not written to the modules. For example, you could make this conditional by adding an input contact or CR to control the instruction with a switch or a programming device. Or, you could add the instruction without any conditions so the selected outputs would be disabled at all times.



STOP Instruction: Sometimes during machine startup you need a way to quickly turn off all the outputs and return to Program Mode. In addition to using the Test Modes and AUX 58 (to configure each individual point), you can also use the STOP instruction. When this instruction is executed, the CPU automatically exits Run Mode and enters Program Mode. Remember, all outputs are turned off during Program Mode. The following diagram shows an example of a condition that returns the CPU to Program Mode.



In the example shown above, you could trigger X20, which would execute the STOP instruction. The CPU would enter Program Mode and all outputs would be turned off.

Run Time Edits

The DL205 CPUs allow you to make changes to the application program during Run Mode. These edits are not "bumpless." Instead, CPU scan is momentarily interrupted (and the outputs are maintained in their current state) until the program change is complete. This means if the output is off, it will remain off until the program change is complete. If the output is on, it will remain on.

WARNING: Only authorized personnel fully familiar with all aspects of the application should make changes to the program. Changes during Run Mode become effective immediately. Make sure you thoroughly consider the impact of any changes to minimize the risk of personal injury or damage to equipment. There are some important operations sequence changes during Run Time Edits.

1. If there is a syntax error in the new instruction, the CPU will not enter the Run Mode.

2. If you delete an output coil reference and the output was on at the time, the output will remain on until it is forced off with a programming device.

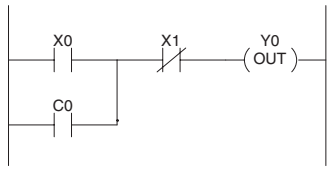
3. Input point changes are not acknowledged during Run Time Edits. So, if you're using a high-speed operation and a critical input comes on, the CPU may not see the change.

Not all instructions can be edited during a Run Time Edit session. The following list shows the instructions that can be edited.

Mnemonic	Description
TMR	Timer
TMRF	Fast timer
TMRA	Accumulating timer
TMRAF	Accumulating fast timer
CNT	Counter
UDC	Up / Down counter
SGCNT	Stage counter
STR, STRN	Store, Store not
AND, ANDN	And, And not
OR, ORN	Or, Or not
STRE, STRNE	Store equal, Store not equal
ANDE, ANDNE	And equal, And not equal
ORE, ORNE	Or equal, Or not equal
STR, STRN	Store greater than or equal Store less than
AND, ANDN	And greater than or equal And less than

Mnemonic	Description
OR, ORN	Or greater than or equal, Or less than
LD	Load data (constant)
LDD	Load data double (constant)
ADDD	Add data double (constant)
SUBD	Subtract data double (constant)
MUL	Multiply (constant)
DIV	Divide (constant)
CMPD	Compare accumulator (constant)
ANDD	And accumulator (constant)
ORD	Or accumulator (constant)
XORD	Exclusive or accumulator (constant)
LDF	Load discrete points to accumulator
OUTF	Output accumulator to discrete points
SHFR	Shift accumulator right
SHFL	Shift accumulator left
NCON	Numeric constant

Use the program logic shown to describe how this process works. In the example, change X0 to C10. Note, the example assumes you have already placed the CPU in Run Mode.



Use the MODE key to select Run Time Edits

MODE NEXT NEXT ENT

```
*MODE CHANGE*
RUN TIME EDIT?
```

Press ENT to confirm the Run Time Edits

ENT (Note, the RUN LED on the DL205 Handheld starts flashing to indicate Run Time Edits are enabled.)

```
*MODE CHANGE*
RUNTIME EDITS
```

Find the instruction you want to change (X0)

SHFT X SET A 0 SHFT FD REF FIND

```
$00000 STR X0
```

Press the arrow key to move to the X. Then enter the new contact (C10).

→ → SHFT C 2 B 1 A 0 ENT

```
RUNTIME EDIT?
STR C10
```

Press ENT to confirm the change

ENT (Note, once you press ENT, the next address is displayed.

```
OR C0
```

Forcing I/O Points

There are many times, especially during machine startup and troubleshooting, where you need the capability to force an I/O point to be either on or off. Before you use a programming device to force any data type, it is important to understand how the DL205 CPUs process the forcing requests.

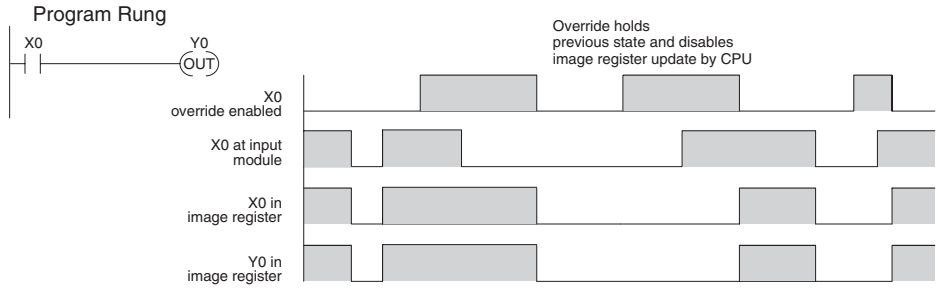
WARNING: Only authorized personnel fully familiar with all aspects of the application should make changes to the program. Make sure you thoroughly consider the impact of any changes to minimize the risk of personal injury or damage to equipment.

Two types of forcing are available with the DL205 CPUs. (Chapter 3 provides a detailed description of how the CPU processes each type of forcing request.)

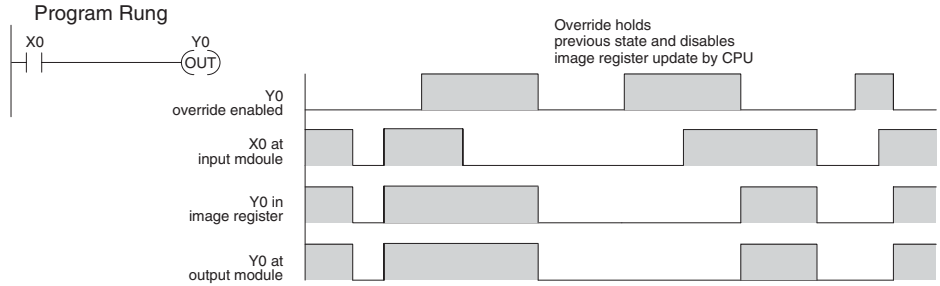
- **Regular Forcing** — This type of forcing can temporarily change the status of a discrete bit. For example, you may want to force an input on, even though it is really off. This allows you to change the point status that was stored in the image register. This value will be valid until the image register location is written to during the next scan. This is primarily useful during testing situations when you need to force a bit on to trigger another event.
- **Bit Override** — (D2-240, D2-250-1, D2-260 or D2-262) Bit override can be enabled on a point-by-point basis by using AUX 59 from the Handheld Programmer or by a menu option in ***Direct***SOFT. You can use Bit Override with X, Y, C, T, CT, and S data types. Bit override basically disables any changes to the discrete point by the CPU. For example, if you enable bit override for X1, and X1 is off at the time, the CPU will not change the state of X1. This means that even if X1 comes on, the CPU will not acknowledge the change. Therefore, if you used X1 in the program, it would always be evaluated as "off" in this case. If X1 was on when the bit override was enabled, then X1 would always be evaluated as "on."

There is an advantage available when you use the bit override feature. The regular forcing is not disabled because the bit override is enabled. For example, if you enabled the Bit Override for Y0 and it was off at the time, the CPU would not change the state of Y0. However, you can still use a programming device to change the status. If you use the programming device to force Y0 on, it will remain on and the CPU will not change the state of Y0. If you then force Y0 off, the CPU will maintain Y0 as off. The CPU will never update the point with the results from the application program or from the I/O update until the bit override is removed from the point.

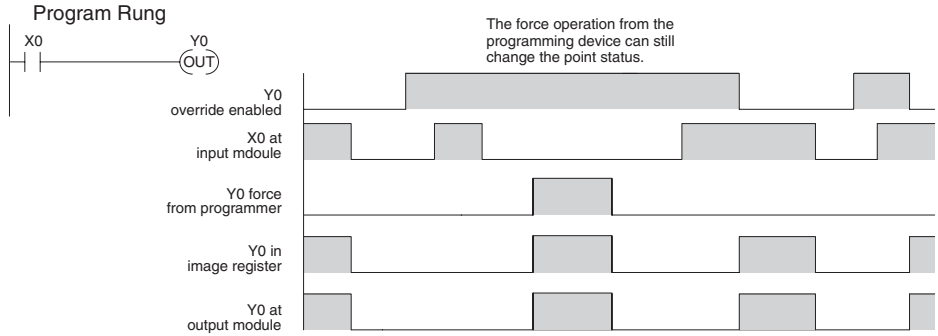
The following diagrams show how the bit override works for both input and output points. The example uses a simple rung, but the concepts are similar for any type of bit memory.



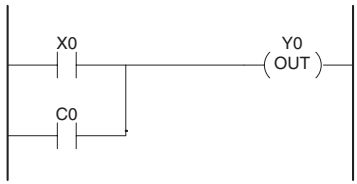
The following diagram shows how the bit override works for an output point. Notice the bit override maintains the output in the current state. If the output is on when the bit override is enabled, then the output stays on. If it is off, then the output stays off.



The following diagram shows how you can use a programming device in combination with the bit override to change the status of the point. Remember, bit override only disables CPU changes. You can still use a programming device to force the status of the point. Plus, since bit override maintains the current status, this enables true forcing. The example shown is for an output point, but you can also use the other bit data types.



The following diagrams show a brief example of how you could use the DL205 Handheld Programmer to force an I/O point. Remember, if you are using the Bit Override feature, the CPU will retain the forced value until you disable the Bit Override or until you remove the force. The image register will not be updated with the status from the input module. Also, the solution from the application program will not be used to update the output image register. The example assumes you have already placed the CPU into Run Mode.



From a clear display, use the following keystrokes

STAT ENT

```
16P STATUS
BIT REF   X
```

Use the PREV or NEXT keys to select the Y data type. (Once the Y appears, press 0 to start at Y0.)

NEXT A 0 ENT

```
 Y   10          Y    0
```

Use arrow keys to select point, then use ON and OFF to change the status

← ← SHFT ON INS

Y2 is now on

```
 Y   10          Y    0
```

Regular Forcing with Direct Access

From a clear display, use the following keystrokes to force Y10 ON

SHFT Y MLS B 1 A 0 SHFT ON INS

Solid fill indicates point is on.

```
BIT FORCE
Y10
```

From a clear display, use the following keystrokes to force Y10 OFF

SHFT Y MLS B 1 A 0 SHFT OFF DEL

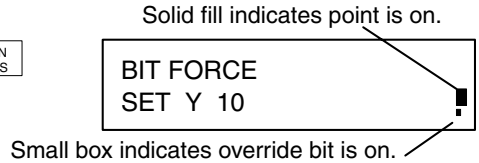
No fill indicates point is off.

```
BIT FORCE
Y10
```

Bit Override Forcing

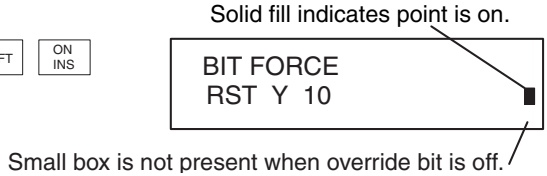
☒ 230
☑ 240
☑ 250-1
☑ 260
☑ 262

From a clear display, use the following keystrokes to turn on the override bit for Y10.



***NOTE**: At this point you can use the PREV and NEXT keys to move to adjacent memory locations and use the SHFT ON keys to set the override bit on.*

From a clear display, use the following keystrokes to turn off the override bit for Y10.

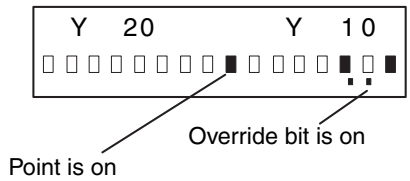


Like the example above, you can use the PREV and NEXT keys to move to adjacent memory locations and use the SHFT OFF keys to set the override bit off.

Bit Override Indicators

Override bit indicators are also shown on the Handheld Programmer status display. Below are the keystrokes to call the status display for Y10 – Y20.

From a clear display, use the following keystrokes to display the status of Y10 – Y20.

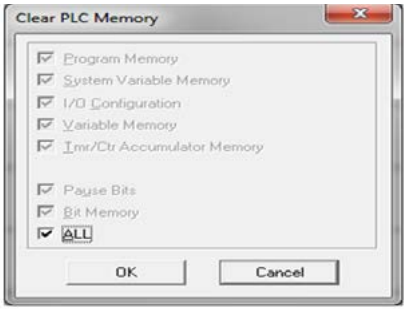


Reset the PLC to Factory Defaults

***NOTE**: Resetting to factory defaults will not clear any password stored in the PLC.*

Resetting a DirectLogic PLC to Factory Defaults is a two-step process. Be sure to have a verified backup of your program using "Save Project to Disk" from the File menu before performing this procedure. Please be aware that the program as well as any settings will be erased and not all settings are stored in the project. In particular you will need to write down any settings for Secondary Communications Ports and manually set the ports up after resetting the PLC to factory defaults.

Step One – While connected to the PLC with DirectSoft, go to the PLC menu and select; "Clear PLC Memory". Check the "ALL" box at the bottom of the list and press "OK".

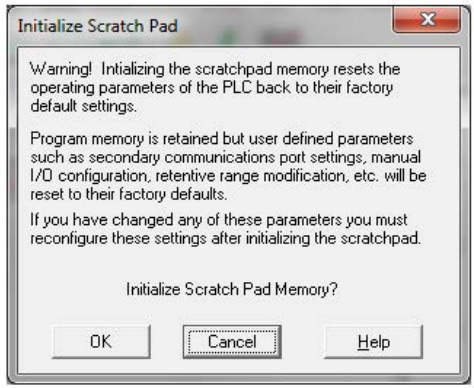


***NOTE 1**: All configurable communications ports will be reset to factory default state. If you are connected via Port 2 or another configurable port, you may be disconnected when this operation is complete.*

***NOTE 2**: Retentive ranges will be reset to the factory settings.*

***NOTE 3**: Manually addressed I/O will be reset to factory default settings.*

Step Two – While connected with DirectSoft, go the PLC menu and then to the "Setup" submenu and select; "Initialize Scratch Pad" and press "Ok



The PLC has now been reset to factory defaults and you can proceed to program the PLC

Auxiliary Functions

In This Appendix...

Introduction

What are Auxiliary Functions?

Many CPU set-up tasks involve the use of Auxiliary (AUX) Functions. The AUX Functions perform many different operations, ranging from clearing ladder memory, displaying the scan time, copying programs to EEPROM in the Handheld Programmer, etc. They are divided into categories that affect different system parameters. You can access the AUX Functions from *Direct*SOFT or from the DL205 Handheld Programmer. The manuals for those products provide step-by-step procedures for accessing the AUX Functions. Some of these AUX Functions are designed specifically for the Handheld Programmer setup, so they will not be needed (or available) with the *Direct*SOFT package. Even though this Appendix provides many examples of how the AUX functions operate, you should supplement this information with the documentation for your choice of programming device. Note, the Handheld Programmer may have additional AUX functions that are not supported with the DL205 CPUs.

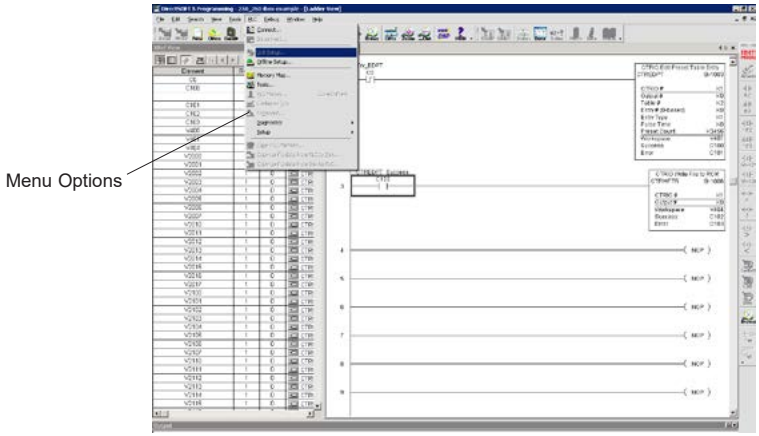
AUX Function and Description		230	240	250-1 260 & 262
AUX 2* — RLL Operations				
21	Check Program	✓	✓	✓
22	Change Reference	✓	✓	✓
23	Clear Ladder Range	✓	✓	✓
24	Clear All Ladders	✓	✓	✓
AUX 3* — V-Memory Operations				
31	Clear V Memory	✓	✓	✓
AUX 4* — I/O Configuration				
41	Show I/O Configuration	✓	✓	✓
42	I/O Diagnostics	✓	✓	✓
44	Power-up I/O Configuration Check	✓	✓	✓
45	Select Configuration	✓	✓	✓
46	Configure I/O	✗	✗	✓
AUX 5* — CPU Configuration				
51	Modify Program Name	✓	✓	✓
52	Display/Change Calendar	✗	✓	✓
53	Display Scan Time	✓	✓	✓
54	Initialize Scratchpad	✓	✓	✓
55	Set Watchdog Timer	✓	✓	✓
56	Set CPU Network Address	✗	✓	✓
57	Set Retentive Ranges	✓	✓	✓
58	Test Operations	✓	✓	✓
59	Bit Override	✗	✓	✓
5B	Counter Interface Config.	✓	✓	✓
5C	Display Error History	✗	✓	✓

AUX Function and Description		230	240	250-1 260 & 262	HPP
AUX 6* — Handheld Programmer Configuration					
61	Show Revision Numbers	✓	✓	✓	-
62	Beeper On / Off	✗	✗	✗	✓
65	Run Self Diagnostics	✗	✗	✗	✓
AUX 7* — EEPROM Operations					
71	Copy CPU memory to HPP EEPROM	✗	✗	✗	✓
72	Write HPP EEPROM to CPU	✗	✗	✗	✓
73	Compare CPU to HPP EEPROM	✗	✗	✗	✓
74	Blank Check (HPP EEPROM)	✗	✗	✗	✓
75	Erase HPP EEPROM	✗	✗	✗	✓
76	Show EEPROM Type (CPU and HPP)	✗	✗	✗	✓
AUX 8* — Password Operations					
81	Modify Password	✓	✓	✓	-
82	Unlock CPU	✓	✓	✓	-
83	Lock CPU	✓	✓	✓	-

✓ supported
✗ not supported
- not applicable

Accessing AUX Functions via DirectSOFT

*Direct*SOFT provides various menu options during both online and offline programming. Some of the AUX functions are only available during online programming, some only during offline programming, and some during both online and offline programming. The following diagram shows an example of the PLC operations menu available within *Direct*SOFT.



Accessing AUX Functions via the Handheld Programmer

You can also access the AUX functions by using a Handheld Programmer. Plus, remember some of the AUX functions are only available from the Handheld Programmer. Sometimes the AUX name or description cannot fit on one display. If you want to see the complete description, press the arrow keys to scroll left and right. Also, depending on the current display, you may have to press CLR more than once.

CLR AUX

```
AUX FUNCTION SELECTION
AUX 2* RLL OPERATIONS
```

Use NEXT or PREV to cycle through the menus

NEXT

```
AUX FUNCTION SELECTION
AUX 3* V OPERATIONS
```

Press ENT to select sub-menus

ENT

```
AUX 3* V OPERATIONS
AUX 31 CLR V-MEMORY
```

You can also enter the exact AUX number to go straight to the sub-menu.

Enter the AUX number directly

CLR D 3 B 1 AUX

```
AUX 3* V OPERATIONS
AUX 31 CLR V-MEMORY
```

AUX 2* — RLL Operations

AUX 21-24

Four AUX functions are available to perform various operations on the control program.

- AUX 21 - Check Program
- AUX 22 - Change Reference
- AUX 23 - Clear Ladder Range
- AUX 24 - Clear Ladders

AUX 21 Check Program

Both the Handheld Programmer and *Direct*SOFT automatically check for errors during program entry. However, there may be occasions when you want to check a program that has already been in the CPU. Two types of checks are available:

- Syntax
- Duplicate References

The Syntax check will find a wide variety of programming errors, such as missing END statements, incomplete FOR/NEXT loops, etc. If you perform this check and get an error, see Appendix B for a complete listing of programming error codes. Correct the problem and then continue running the Syntax check until the message "NO SYNTAX ERROR" appears.

Use the Duplicate Reference check to verify you have not used the same output coil reference more than once. Note, this AUX function will also find the same outputs even if they have been used with the OROUT instruction, which is perfectly acceptable.

This AUX function is available on the PLC Diagnostics sub-menu from within *Direct*SOFT.

AUX 22 Change Reference

There will be times when you need to change an I/O address reference or control relay reference. AUX 22 allows you to quickly and easily change all occurrences, (within an address range), of a specific instruction. For example, you can replace every instance of X5 with X10.

AUX 23 Clear Ladder Range

There have been many times when you take existing programs and add or remove certain portions to solve new application problems. By using AUX 23 you can select and delete a portion of the program. *Direct*SOFT does not have a menu option for this AUX function, but you can select the appropriate portion of the program and cut it with the editing tools.

AUX 24 Clear Ladders

AUX 24 clears the entire program from CPU memory. Before you enter a new program, you should always clear ladder memory. This AUX function is available on the PLC > Clear PLC sub-menu within *Direct*SOFT.

AUX 3* — V-memory Operations

AUX 31

- AUX 31 - Clear V-memory

AUX 31 Clear V-Memory

AUX 31 clears all the information from the V-memory locations available for general use. This AUX function is available on the PLC > Clear PLC sub-menu within *Direct*SOFT.

AUX 4* — I/O Configuration

AUX 41-46

Several AUX functions are available to set up, view, or change the I/O configuration.

- AUX 41 — Show I/O Configuration
- AUX 42 — I/O Diagnostics
- AUX 44 — Power-up Configuration Check
- AUX 45 — Select Configuration
- AUX 46 — Configure I/O

AUX 41 Show I/O Configuration

This AUX function allows you to display the current I/O configuration. With the Handheld Programmer, you can scroll through each base and I/O slot to view the complete configuration. The configuration shows the type of module installed in each slot. *Direct*SOFT provides the same information, but it is much easier to view because you can view a complete base on one screen.

AUX 42 I/O Diagnostics

This is one of the most useful AUX functions available in the DL205 system. This AUX function will show you the exact base and slot location of any I/O module error that has occurred. This feature is also available within *Direct*SOFT under the PLC > Diagnostics sub-menu.

AUX 44 Power-up Configuration Check

Select this feature to quickly detect any changes that may have occurred while the power was disconnected. For example, if someone placed an output module in a slot that previously held an input module, the configuration check would detect the change.

If the system detects a change in the I/O configuration at power-up, an error code E252 NEW I/O CONFIGURATION will be generated. You can use AUX 42 to determine the exact base and slot location where the change occurred. This feature is also available within *Direct*SOFT under the PLC > Setup sub-menu.

WARNING: You should always correct any I/O configuration errors before you place the CPU into RUN mode. Uncorrected errors can cause unpredictable machine operation that can result in a risk of personal injury or damage to equipment.

AUX 45 Select Configuration

Even though the CPU can automatically detect configuration changes, you may actually want the new I/O configuration to be used. For example, you may have intentionally changed a module to use with a new program. You can use AUX 45 to select the new configuration, or, keep the existing configuration that is stored in memory. This feature is also available within *Direct*SOFT from the PLC > Setup sub-menu.

WARNING: Make sure the I/O configuration being selected will work properly with the CPU program. You should always correct any I/O configuration errors before you place the CPU into RUN mode. Uncorrected errors can cause unpredictable machine operation that can result in a risk of personal injury or damage to equipment.

AUX 46 Configure I/O

The D2-250–1, D2-260 and D2-262 CPUs allow you to use AUX 46 to manually assign I/O addresses for any or all I/O slots on the local or expansion bases. It is generally much easier to do the I/O configuration operations from within *Direct*SOFT. The software package provides a really nice screen that is available from the PLC > Configure I/O sub-menu.

This feature is useful if you have a standard configuration you must sometimes change slightly to accommodate special requests. For example, you may require two adjacent input modules to have addresses starting at X10 and X200 respectively.

In automatic configuration, the addresses were assigned on 8-point boundaries. Manual configuration assumes that all modules are at least 16 points, so you can only assign addresses that are a multiple of 20 (octal). For example, X30 and Y50 would not be valid starting addresses for a module. X20 and Y40 are valid examples of starting addresses in a manual configuration. This does not mean you can only use 16- or 32-point modules with manual configuration. You can use 8-point modules, but 16 addresses will be assigned and 8 of them are unused.

WARNING: If you manually configure an I/O slot, the I/O addressing for the other modules will change. This is because the DL205 products do not allow you to assign duplicate I/O addresses. You should always correct any I/O configuration errors before you place the CPU into RUN mode. Uncorrected errors can cause unpredictable machine operation that can result in a risk of personal injury or damage to equipment.

Once you have manually configured the addresses for an I/O slot, the system will automatically retain these values even after a power cycle. You can remove any manual configuration changes by simply performing an automatic configuration.

AUX 5* — CPU Configuration

AUX 51-5C

Several AUX functions are available to set up, view, or change the CPU configuration.

- AUX 51 — Modify Program Name
- AUX 52 — Display / Change Calendar
- AUX 53 — Display Scan Time
- AUX 54 — Initialize Scratchpad
- AUX 55 — Set Watchdog Timer
- AUX 56 — CPU Network Address
- AUX 57 — Set Retentive Ranges
- AUX 58 — Test Operations
- AUX 59 — Bit Override
- AUX 5B — Counter Interface Configuration
- AUX 5C — Display Error / Message History

AUX 51 Modify Program Name

The DL205 products can use a program name for the CPU program or a program stored on EEPROM in the Handheld Programmer. Note, you cannot have multiple programs stored on the EEPROM. The program name can be up to 8 characters in length and can use any of the available characters (A–Z, 0–9). AUX 51 allows you to enter a program name. You can also perform this operation from within ***Direct***SOFT by using the PLC > Setup sub-menu. Once you've entered a program name, you can only clear the name by using AUX 54 to reset the system memory. Make sure you understand the possible ramifications of AUX 54 before you use it!

AUX 52 Display/Change Calendar

The D2-240, D2-250–1, D2-260 and D2-262 CPUs have a clock and calendar feature. If you are using this, you can use the Handheld Programmer and AUX 52 to set the time and date. The following format is used.

- Date — Year, Month, Date, Day of week (0 – 6, Sunday through Saturday)
- Time — 24-hour format, Hours, Minutes, Seconds

You can use the AUX function to change any component of the date or time. However, the CPU will not automatically correct any discrepancy between the date and the day of the week. For example, if you change the date to the 15th of the month and the 15th is on a Thursday, you will also have to change the day of the week (unless the CPU already shows the date as Thursday).

You can also perform this operation from within ***Direct***SOFT by using the PLC > Setup sub-menu.

AUX 53 Display Scan Time

AUX 53 displays the current, minimum, and maximum scan times. The minimum and maximum times are the ones that have occurred since the last Program Mode to Run Mode transition. You can also perform this operation from within *Direct*SOFT by using the PLC > Diagnostics sub-menu.

AUX 54 Initialize Scratchpad

The DL205 CPUs maintain system parameters in a memory area often referred to as the "scratchpad." In some cases, you may make changes to the system setup that will be stored in system memory. For example, if you specify a range of Control Relays (CR) as retentive, these changes are stored.

NOTE: *You may never have to use this feature unless you have made changes that affect system memory. Usually, you'll only need to initialize the system memory if you are changing programs and the old program required a special system setup. You can usually change from program to program without ever initializing system memory.*

AUX 54 resets the system memory to the default values. You can also perform this operation from within *Direct*SOFT by using the PLC > Setup sub-menu.

AUX 55 Set Watchdog Timer

The DL205 CPUs have a "watchdog" timer that is used to monitor the scan time. The default value set from the factory is 200ms. If the scan time exceeds the watchdog time limit, the CPU automatically leaves RUN mode and enters PGM mode. When the scan overrun occurs, the Handheld Programmer displays the following message: E003 S/W TIMEOUT.

Use AUX 55 to increase or decrease the watchdog timer value. You can also perform this operation from within *Direct*SOFT by using the PLC > Setup sub-menu.

AUX 56 CPU Network Address

Since the D2-240, D2-250–1, D2-260 and D2-262 CPUs have an additional communication port, you can use the Handheld Programmer to set the network address for the port and the port communication parameters. The default settings are:

- Station address 1
- HEX mode
- Odd parity

You can use this port with either the Handheld Programmer, *Direct*SOFT, or, as a *Direct*NET communication port. The *Direct*NET Manual provides additional information about communication settings required for network operation.

NOTE: *You will only need to use this procedure if you have the bottom port connected to a network. Otherwise, the default settings will work fine.*

Use AUX 56 to set the network address and communication parameters. You can also perform this operation from within *Direct*SOFT by using the PLC > Setup sub-menu.

AUX 57 Set Retentive Ranges

The DL205 CPUs provide certain ranges of retentive memory by default. The default ranges are suitable for many applications, but you can change them if your application requires additional retentive ranges or no retentive ranges at all. The default settings are:

Memory Area	D2-230		D2-240		D2-250-1		D2-260/D2-262	
	Default Range	Available Range	Default Range	Available Range	Default Range	Available Range	Default Range	Available Range
Control Relays	C300-C377	C0-C377	C300-C377	C0-C377	C1000-C1777	C0-C1777	C1000-C1777	C0-C3777
V-Memory	V2000-V7777	V0-V7777	V2000-V7777	V0-V7777	V1400-V3777	V0-V17777	V1400-V3777	V0-V37777
Timers	None by default	T0-T77	None by default	T0-T177	None by default	T0-T377	None by default	T0-T377
Counters	CT0-CT77	CT0-CT77	CT0-CT177	CT0-CT177	CT0-CT177	CT0-CT177	CT0-CT177	CT0-CT377
Stages	None by default	S0-S377	None by default	S0-S777	None by default	S0-S1777	None by default	S0-S1777

Use AUX 57 to change the retentive ranges. You can also perform this operation from within *Direct*SOFT by using the PLC > Setup sub-menu.

WARNING: The DL205 CPUs do not come with a battery. They have a super capacitor which will help in retaining values in the event of a power loss. The retention time varies depending on environmental conditions but could possibly last as long as 1 week (retention in the D2-262 is 1.9 hours, maximum). If the retentive ranges are important for your application the optional battery must be used.

AUX 58 Test Operations

In normal Run Mode, the outputs are turned off when you return to Program Mode. In TEST-RUN mode you can set each individual output to either turn off, or, hold its last output state on the transition to TEST-PGM mode. The ability to hold the output states is especially useful, since it allows you to maintain key system I/O points for examination. See Chapter 9 for a description of the Test Modes.

You can use AUX 58 to configure each individual output. You can also perform this operation from within *Direct*SOFT by using the PLC > Setup sub-menu.

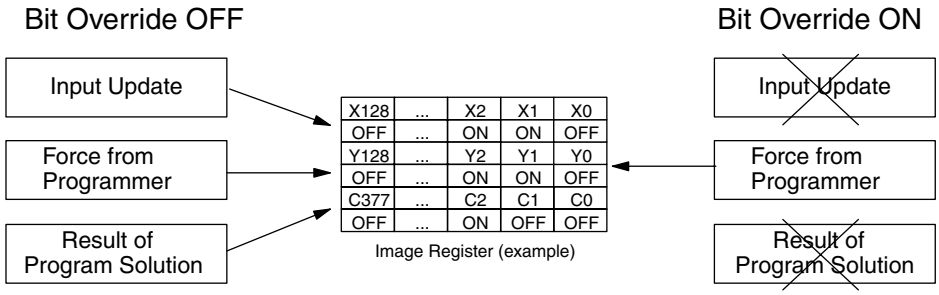
AUX 59 Bit Override

Bit override can be enabled on a point-by-point basis by using AUX 59 from the Handheld Programmer or, by a menu option from within *Direct*SOFT. Bit override basically disables any changes to the discrete point by the CPU. For example, if you enable bit override for X1, and X1 is off at the time, then the CPU will not change the state of X1. This means that even if X1 comes on, the CPU will not acknowledge the change. So, if you used X1 in the program, it would always be evaluated as "off" in this case. Of course, if X1 was on when the bit override was enabled, then X1 would always be evaluated as "on."

***NOTE: Direct**Net protocol does not support single-bit write operations.*

There is an advantage available when you use the bit override feature. The regular forcing is not disabled because the bit override is enabled. For example, if you enabled the Bit Override for Y0 and it was off at the time, then the CPU would not change the state of Y0. However, you can still use a programming device to change the status. Now, if you use the programming device to force Y0 on, it will remain on and the CPU will not change the state of Y0. If you then force Y0 off, the CPU will maintain Y0 as off. The CPU will never update the point with the results from the application program or from the I/O update until the bit override is removed from the point.

The following diagram shows a brief overview of the bit override feature. Notice the CPU does not update the Image Register when bit override is enabled.



AUX 5B Counter Interface Configuration

AUX 5B is used with the DL205 Counter Interface module D2-CTRINT to select the module configuration. You can choose the type of counter, set the counter parameters, etc. See the DL205 Counter Interface Module manual for a complete description of how to select the various counter features.

AUX 5C Display Error History

The D2-240, D2-250–1, D2-260 and D2-262 CPUs will automatically log any system error codes and custom messages created with the FAULT instructions. The CPU logs the error code, date, and time the error occurred. There are two separate tables that store this information.

- Error Code Table – the system logs up to 32 errors in the table. When an error occurs, the errors already on the table are pushed down and the most recent error is loaded into the top slot. If the table is full when an error occurs, the oldest error is pushed out (erased) of the table.
- Message Table – the system logs up to 16 messages in this table. When a message is triggered, the messages already stored in the table are pushed down and the most recent message is loaded into the top slot. If the table is full when an error occurs, the oldest message is pushed out (erased) of the table.

The following is an example of an error table for messages.

Date	Time	Message
2008-05-26	08:41:51:11	*Conveyor - 2 stopped
2008-04-30	17:01:11:56	*Conveyor - 1 stopped
2008-04-30	17:01:11:12	*Limit SW1 failed
2008-04-28	03:25:14:31	*Saw Jam Detect

You can use AUX Function 5C to show the error codes or messages. You can also view the errors and messages from within ***Direct***SOFT by using the PLC > Diagnostics sub-menu.

AUX 6* — Handheld Programmer Configuration

AUX 61, 62 and 65

Several AUX functions are available that you can use to set up, view, or change the Handheld Programmer configuration.

- AUX 61 — Show Revision Numbers
- AUX 62 — Beeper On/Off
- AUX 65 — Run Self Diagnostics

AUX 61 Show Revision Numbers

As with most industrial control products, there are cases when additional features and enhancements are made. Sometimes these new features only work with certain releases of firmware. By using AUX 61, you can quickly view the CPU and Handheld Programmer firmware revision numbers. This information (for the CPU) is also available from within *Direct*SOFT from the PLC > Diagnostics sub-menu.

AUX 62 Beeper On/Off

The Handheld Programmer has a beeper that provides confirmation of keystrokes. You can use Auxiliary (AUX) Function 62 to turn the beeper OFF or ON.

AUX 65 Run Self Diagnostics

If you think the Handheld Programmer is not operating correctly, you can use AUX 65 to run a self-diagnostics program. You can check the following items.

- Keypad
- Display
- LEDs and Backlight
- Handheld Programmer EEPROM check

AUX 7* — EEPROM Operations

AUX 71 – 76

Several AUX functions are available to move programs between the CPU memory and an optional EEPROM installed in the Handheld Programmer.

- AUX 71 — Read from CPU memory to HPP EEPROM
- AUX 72 — Write HPP EEPROM to CPU
- AUX 73 — Compare CPU to HPP EEPROM
- AUX 74 — Blank Check (HPP EEPROM)
- AUX 75 — Erase HPP EEPROM
- AUX 76 — Show EEPROM Type (CPU and HPP)

Transferable Memory Areas

Many of these AUX functions allow you to copy different areas of memory to and from the CPU and Handheld Programmer. The following table shows the areas that may be mentioned.

Option and Memory Type	D2-240 Default Range	D2-230 Default Range
1:PGM — Program	$00000–$02559	$00000–$02047
2:V — V-memory	$00000–$4777	$00000–$04777
3:SYS — System	Non-selectable copies system parameters	
4:etc (All)—Program, System and *non-volatile* V-memory	Non-selectable	Non-selectable

AUX 71 CPU to HPP EEPROM

AUX 71 copies information from the CPU memory to an EEPROM installed in the Handheld Programmer. You can copy different types of information from CPU memory as shown in the previous table.

AUX 72 HPP EEPROM to CPU

AUX 72 copies information from an EEPROM installed in the Handheld Programmer to the CPU. You can copy different portions of EEPROM (HP) memory to the CPU memory as shown in the previous table. The amount of data you can copy depends on the CPU.

AUX 73 Compare HPP EEPROM to CPU

AUX 73 compares the program in the Handheld Programmer (EEPROM) with the CPU program. You can compare different types of information as shown previously. There is also an option called "etc." that allows you to check all of the areas sequentially without re-executing the AUX function every time.

AUX 74 HPP EEPROM Blank Check

AUX 74 allows you to check the EEPROM in the Handheld Programmer to make sure it is blank. It's a good idea to use this function any time you start to copy an entire program to an EEPROM in the Handheld Programmer.

AUX 75 Erase HPP EEPROM

AUX 75 allows you to clear all data in the EEPROM in the Handheld Programmer. You should use this AUX function before you copy a program from the CPU.

AUX 76 Show EEPROM Type

You can use AUX 76 to quickly determine what size EEPROM is installed in the CPU and Handheld Programmer. The D2-230 and D2-240 use different size EEPROMs. See Chapter 3 for additional information.

AUX 8* — Password Operations

AUX 81 - 83

AUX functions are available to modify or enable the CPU password. You can use these features during on-line communications with the CPU, or, you can also use them with an EEPROM installed in the Handheld Programmer during off-line operation. This will allow you to develop a program in the Handheld Programmer and include password protection.

- AUX 81 — Modify Password
- AUX 82 — Unlock CPU
- AUX 83 — Lock CPU

AUX 81 Modify Password

You can use AUX 81 to provide an extra measure of protection by entering a password that prevents unauthorized machine operations. The password must be an eight-character numeric (0–9) code. Once you've entered a password, you can remove it by entering all zeros (00000000). This is the default from the factory.

Once you've entered a password, you can lock the CPU against access. There are two ways to lock the CPU with the Handheld Programmer.

- The CPU is always locked after a power cycle (if a password is present).
- You can use AUX 83 and AUX 82 to lock and unlock the CPU.

You can also enter or modify a password from within ***Direct***SOFT by using the PLC > Password sub-menu. This feature works slightly differently in ***Direct***SOFT. Once you've entered a password, the CPU is automatically locked when you exit the software package. It will also be locked if the CPU is power cycled.

WARNING: Make sure you remember the password before you lock the CPU. Once the CPU is locked you cannot view, change, or erase the password. If you do not remember the password, you have to return the CPU to the factory for password removal (For US customers only).

NOTE: *The D2–240, D2-250–1, D2–260 and D2-262 CPUs support multi-level password protection of the ladder program. This allows password protection while not locking the communication port to an operator interface. The multi-level password can be invoked by creating a password with an upper case "A" followed by seven numeric characters (e.g., A1234567).*

AUX 82 Unlock CPU

AUX 82 can be used to unlock a CPU that has been password protected. ***Direct***SOFT will automatically ask you to enter the password if you attempt to communicate with a CPU that contains a password.

AUX 83 Lock CPU

AUX 83 can be used to lock a CPU that contains a password. Once the CPU is locked, you will have to enter a password to gain access. Remember, this is not necessary with ***Direct***SOFT since the CPU is automatically locked whenever you exit the software package.

DL205 Error Codes

In This Appendix...

DL205 Error Codes

DL205 Error Code	Description
E003 SOFTWARE TIME-OUT	If the program scan time exceeds the time allotted to the watchdog timer, this error will occur. SP51 will be on and the error code will be stored in V7755. To correct this problem add RSTWT instructions in FOR NEXT loops and subroutines or use AUX 55 to extend the time allotted to the watchdog timer.
E041 CPU BATTERY LOW	The CPU battery is low and should be replaced. SP43 will be on and the error code will be stored in V7757.
E099 PROGRAM MEMORY EXCEEDED	If the compiled program length exceeds the amount of available CPU RAM, this error will occur. SP52 will be on and the error code will be stored in V7755. Reduce the size of the application program.
E104 WRITE FAILED	A write to the CPU was not successful. Disconnect the power, remove the CPU, and make sure the EEPROM is not write protected. If the EEPROM is not write protected, make sure the EEPROM is installed correctly. If both conditions are OK, replace the CPU.
E151 BAD COMMAND	A parity error has occurred in the application program. SP44 will be on and the error code will be stored in V7755. This problem may possibly be due to electrical noise. Clear the memory and download the program again. Correct any grounding problems. If the error returns, replace the EEPROM or the CPU.
E155 RAM FAILURE	A checksum error has occurred in the system RAM. SP44 will be on and the error code will be stored in V7755. This problem may possibly be due to a low battery, electrical noise or a CPU RAM failure. Clear the memory and download the program again. Correct any grounding problems. If the error returns, replace the CPU.
E202 MISSING I/O MODULE	An I/O module has failed to communicate with the CPU or is missing from the base. SP45 will be on and the error code will be stored in V7756. Run AUX42 to determine the slot and base location of the module reporting the error.
E210 POWER FAULT	A short duration power drop-out occurred on the main power line supplying power to the base.
E250 COMMUNICATION FAILURE IN THE I/O CHAIN	A failure has occurred in the local I/O system. The problem could be in the base I/O bus or the base power supply. SP45 will be on and the error code will be stored in V7755. Run AUX42 to determine the base location reporting the error.
E252 NEW I/O CFG	This error occurs when the auto configuration check is turned on in the CPU and the actual I/O configuration has changed either by moving modules in a base or changing types of modules in a base. You can return the modules to the original position/types or run AUX45 to accept the new configuration. SP47 will be on and the error code will be stored in V7755.
E262 I/O OUT OF RANGE	An out-of-range I/O address has been encountered in the application program. Correct the invalid address in the program. SP45 will be on and the error code will be stored in V7755.
E312 HP COMM ERROR 2	A data error was encountered during communications with the CPU. Clear the error and retry the request. If the error continues, check the cabling between the two devices, replace the Handheld Programmer, then if necessary replace the CPU. SP46 will be on and the error code will be stored in V7756.

DL205 Error Code	Description
E313 HP COMM ERROR 3	An address error was encountered during communications with the CPU. Clear the error and retry the request. If the error continues, check the cabling between the two devices; replace the Handheld Programmer; then, if necessary, replace the CPU. SP46 will be on and the error code will be stored in V7756.
E316 HP COMM ERROR 6	A mode error was encountered during communications with the CPU. Clear the error and retry the request. If the error continues, replace the Handheld Programmer; then if necessary, replace the CPU. SP46 will be on and the error code will be stored in V7756.
E320 HP COMM TIME-OUT	The CPU did not respond to the Handheld Programmer communication request. Check to ensure cabling is correct and not defective. Power cycle the system; If the error continues, replace the CPU first and then the Handheld Programmer, if necessary.
E321 COMM ERROR	A data error was encountered during communication with the CPU. Check to ensure cabling is correct and not defective. Power cycle the system and, if the error continues, replace the CPU first and then the Handheld Programmer, if necessary.
E4** NO PROGRAM	A syntax error exists in the application program. The most common is a missing END statement. Run AUX21 to determine which one of the E4** series of errors is being flagged. SP52 will be on and the error code will be stored in V7755.
E401 MISSING END STATEMENT	All application programs must terminate with an END statement. Enter the END statement in the appropriate location in your program. SP52 will be on and the error code will be stored in V7755.
E402 MISSING LBL	A GOTO, GTS, MOVMC or LDLBL instruction was used without the appropriate label. Refer to the programming manual for details on these instructions. SP52 will be on and the error code will be stored in V7755.
E403 MISSING RET (D2-240 ONLY)	A subroutine in the program does not end with the RET instruction. SP52 will be on and the error code will be stored in V7755.
E404 MISSING FOR (D2-240, D2-250-1, D2-260, D2-262)	A NEXT instruction does not have the corresponding FOR instruction. SP52 will be on and the error code will be stored in V7755.
E405 MISSING NEXT (D2-240, D2-250-1, D2-260, D2-262)	A FOR instruction does not have the corresponding NEXT instruction. SP52 will be on and the error code will be stored in V7755.
E406 MISSING IRT	An interrupt routine in the program does not end with the IRT instruction. SP52 will be on and the error code will be stored in V7755.
E412 SBR/LBL>64 (D2-240, D2-250-1, D2-260, D2-262)	There are more than 64 SBR, LBL or DLBL instructions in the program. This error is also returned if there is greater than 128 GTS or GOTO instructions used in the program. SP52 will be on and the error code will be stored in V7755.

DL205 Error Code	Description
E413 FOR/NEXT>64 (D2-240, D2-250–1, D2-260, D2-262)	There are more than 64 FOR/NEXT loops in the application program. SP52 will be on and the error code will be stored in V7755.
E421 DUPLICATE STAGE REFERENCE	Two or more SG or ISG labels exist in the application program with the same number. A unique number must be allowed for each Stage and Initial Stage. SP52 will be on and the error code will be stored in V7755.
E422 DUPLICATE SBR/LBL REFERENCE	Two or more SBR or LBL instructions exist in the application program with the same number. A unique number must be allowed for each Subroutine and Label. SP52 will be on and the error code will be stored in V7755.
E423 NESTED LOOPS (D2-240, D2-250–1, D2-260, D2-262)	Nested loops (programming one FOR/NEXT loop inside of another) is not allowed in the D2-240, D2-250–1, D2-260, or D2-262 series. SP52 will be on and the error code will be stored in V7755.
E431 INVALID ISG/SG ADDRESS	An ISG or SG must not be programmed after the end statement such as in a subroutine. SP52 will be on and the error code will be stored in V7755.
E432 INVALID JUMP (GOTO) ADDRESS (D2-240, D2-250–1, D2-260, D2-262)	A LBL that corresponds to a GOTO instruction must not be programmed after the end statement such as in a subroutine. SP52 will be on and the error code will be stored in V7755.
E433 INVALID SBR ADDRESS (D2-240, D2-250–1, D2-260, D2-262)	A SBR must be programmed after the end statement, not in the main body of the program or in an interrupt routine. SP52 will be on and the error code will be stored in V7755.
E435 INVALID RT ADDRESS (D2-240, D2-250–1, D2-260, D2-262)	A RT must be programmed after the end statement, not in the main body of the program or in an interrupt routine. SP52 will be on and the error code will be stored in V7755.
E436 INVALID INT ADDRESS	An INT must be programmed after the end statement, not in the main body of the program. SP52 will be on and the error code will be stored in V7755.
E438 INVALID IRT ADDRESS	An IRT must be programmed after the end statement, not in the main body of the program. SP52 will be on and the error code will be stored in V7755.
E440 INVALID DATA ADDRESS	Either the DLBL instruction has been programmed in the main program area (not after the END statement), or the DLBL instruction is on a rung containing input contact(s).
E441 ACON/NCON (D2-240, D2-250–1, D2-260, D2-262)	An ACON or NCON must be programmed after the end statement, not in the main body of the program. SP52 will be on and the error code will be stored in V7755.

DL205 Error Codes	Description
E451 BAD MLS/MLR	MLS instructions must be numbered in ascending order from top to bottom.
E452 X AS COIL	An X data type is being used as a coil output.
E453 MISSING T/C	A timer or counter contact is being used where the associated timer or counter does not exist.
E454 BAD TMRA	One of the contacts is missing from a TMRA instruction.
E455 BAD CNT	One of the contacts is missing from a CNT or UDC instruction.
E456 BAD SR	One of the contacts is missing from the SR instruction.
E461 STACK OVERFLOW	More than nine levels of logic have been stored on the stack. Check the use of OR STR and AND STR instructions.
E462 STACK UNDERFLOW	An unmatched number of logic levels have been stored on the stack. Ensure the number of AND STR and OR STR instructions match the number of STR instructions.
E463 LOGIC ERROR	A STR instruction was not used to begin a rung of ladder logic.
E464 MISSING CKT	A rung of ladder logic is not terminated properly.
E471 DUPLICATE COIL REFERENCE	Two or more OUT instructions reference the same I/O point.
E472 DUPLICATE TMR REFERENCE	Two or more TMR instructions reference the same number.

DL205 Error Code	Description
E473 DUPLICATE CNT REFERENCE	Two or more CNT instructions reference the same number.
E480 INVALID CV ADDRESS	The CV instruction is used in a subroutine or program interrupt routine. The CV instruction may only be used in the main program area (before the END statement).
E481 CONFLICTING INSTRUCTIONS	An instruction exists between convergence stages.
E482 MAX. CV INSTRUCTIONS EXCEEDED	Number of CV instructions exceeds 17.
E483 INVALID CVJMP ADDRESS	CVJMP has been used in a subroutine or a program interrupt routine.
E484 MISSING CV INSTRUCTION	CVJMP is not preceded by the CV instruction. A CVJMP must immediately follow the CV instruction.
E485 NO CVJMP	A CVJMP instruction is not placed between the CV and the SG, ISG, BLK, BEND, END instruction.
E486 INVALID BCALL ADDRESS	A BCALL is used in a subroutine or a program interrupt routine. The BCALL instruction may only be used in the main program area (before the END statement).
E487 MISSING BLK INSTRUCTION	The BCALL instruction is not followed by a BLK instruction.
E488 INVALID BLK ADDRESS	The BLK instruction is used in a subroutine or a program interrupt. Another BLK instruction is used between the BCALL and the BEND instructions.
E489 DUPLICATED CR REFERENCE	The control relay used for the BLK instruction is being used as an output elsewhere.
E490 MISSING SG INSTRUCTION	The BLK instruction is not immediately followed by the SG instruction.

DL205 Error Code	Description
E491 INVALID ISG INSTRUCTION ADDRESS	There is an ISG instruction between the BLK and BEND instructions.
E492 INVALID BEND ADDRESS	The BEND instruction is used in a subroutine or a program interrupt routine. The BEND instruction is not followed by a BLK instruction.
E493 MISSING REQUIRED INSTRUCTION	A [CV, SG, ISG, BLK, BEND] instruction must immediately follow the BEND instruction.
E494 MISSING BEND INSTRUCTION	The BLK instruction is not followed by a BEND instruction.
E499 PRINT INSTRUCTION	Invalid PRINT instruct usage. Quotations and/or spaces were not entered or entered incorrectly.
E501 BAD ENTRY	An invalid keystroke or series of keystrokes was entered into the Handheld Programmer.
E502 BAD ADDRESS	An invalid or out-of-range address was entered into the Handheld Programmer.
E503 BAD COMMAND	An invalid instruction was entered into the Handheld Programmer.
E504 BAD REF/VAL	An invalid value or reference number was entered with an instruction.
E505 INVALID INSTRUCTION	An invalid instruction was entered into the Handheld Programmer.
E506 INVALID OPERATION	An invalid operation was attempted by the Handheld Programmer.
E520 BAD OP–RUN	An operation which is invalid in the RUN mode was attempted by the Handheld Programmer.

DL205 Error Code	Description
E521 BAD OP–TRUN	An operation which is invalid in the TEST RUN mode was attempted by the Handheld Programmer.
E523 BAD OP–TPGM	An operation which is invalid in the TEST PROGRAM mode was attempted by the Handheld Programmer.
E524 BAD OP–PGM	An operation which is invalid in the PROGRAM mode was attempted by the Handheld Programmer.
E525 MODE SWITCH (D2-240, D2-250–1, D2-260, D2-262)	An operation was attempted by the Handheld Programmer while the CPU mode switch was in a position other than the TERM position.
E526 OFF LINE	The Handheld Programmer is in the OFFLINE mode. To change to the ONLINE mode, use the MODE key.
E527 ON LINE	The Handheld Programmer is in the ONLINE mode. To change to the OFFLINE mode, use the MODE key.
E528 CPU MODE	The operation attempted is not allowed during a Run Time Edit.
E540 CPU LOCKED	The CPU has been password locked. To unlock the CPU, use AUX82 with the password.
E541 WRONG PASSWORD	The password used to unlock the CPU with AUX82 was incorrect.
E542 PASSWORD RESET	The CPU powered up with an invalid password and reset the password to 00000000. A password may be re-entered using AUX81.
E601 MEMORY FULL	Attempted to enter an instruction which required more memory than is available in the CPU.
E602 INSTRUCTION MISSING	A search function was performed and the instruction was not found.
E604 REFERENCE MISSING	A search function was performed and the reference was not found.

DL205 Error Code	Description
E610 BAD I/O TYPE	The application program has referenced an I/O module as the incorrect type of module.
E620 OUT OF MEMORY	An attempt to transfer more data between the CPU and Handheld Programmer than the receiving device can hold. DV-1000: mismatch between MOVMC and LBL instructions.
E621 EEPROM NOT BLANK	An attempt to write to a non-blank EEPROM was made. Erase the EEPROM and then retry the write.
E622 NO HPP EEPROM	A data transfer was attempted with no EEPROM (or possibly a faulty EEPROM) installed in the Handheld Programmer.
E623 SYSTEM EEPROM	A function was requested with an EEPROM which contains system information only.
E624 V-MEMORY ONLY	A function was requested with an EEPROM which contains V-memory data only.
E625 PROGRAM ONLY	A function was requested with an EEPROM which contains program data only.
E627 BAD WRITE	An attempt to write to a write protected or faulty EEPROM was made. Check the write protect jumper and replace the EEPROM if necessary.
E628 EEPROM TYPE ERROR	The wrong size EEPROM is being used. The D2-230 and D2-240 CPUs use different size EEPROMs.
E640 COMPARE ERROR	A compare between the EEPROM and the CPU was found to be in error.
E650 HPP SYSTEM ERROR	A system error has occurred in the Handheld Programmer. Power cycle the Handheld Programmer. If the error returns, replace the Handheld Programmer.
E651 HPP ROM ERROR	A ROM error has occurred in the Handheld Programmer. Power cycle the Handheld Programmer. If the error returns, replace the Handheld Programmer.
E652 HPP RAM ERROR	A RAM error has occurred in the Handheld Programmer. Power cycle the Handheld Programmer. If the error returns, replace the Handheld Programmer.

Instruction Execution Times

In This Appendix...

Introduction

This appendix contains several tables that provide the instruction execution times for the DL205 CPUs. One thing you will notice is that many of the execution times depend on the type of data being used with the instruction. For example, you'll notice that some of the instructions that use V-memory locations are further defined by the following items:

- Data Registers
- Bit Registers

V-memory Data Registers

Some V-memory locations are considered data registers. For example, the V-memory locations that store the timer or counter current values, or just regular user V-memory, would be considered as a V-memory data register. Don't think that you cannot load a bit pattern into these types of registers, you can. It's just that their primary use is as a data register. The following locations are considered as data registers.

Data Registers	D2-230	D2-240	D2-250-1	D2-260/D2-262
Timer Current Values	V0 - V77	V0 - V177	V0 - V377	V0 - V377
Counter Current Values	V1000 - V1077	V1000 - V1177	V1000 - V1177	V1000 - V1377
User Data Words	V2000 - V2377 V4000 - V4177	V2000 - V3777 V4000 - V4377	V1400 - V7377 V10000 - V17777	V400 - V777 V1400 - V7377 V10000 - V35777

V-Memory Bit Registers

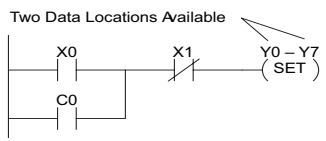
You may recall that some of the discrete points such as X, Y, C, etc., are mapped automatically into V-memory. The following locations that contain this data are considered bit registers.

Bit Registers	D2-230	D2-240	D2-250-1	D2-260/D2-262
Input Points (X)	V40400 - V40407	V40400 - V40423	V40400 - V40437	V40400 - V40477
Output Points (Y)	V40500 - V40507	V40500 - V40523	V40500 - V40537	V40500 - V40577
Control Relays (C)	V40600 - V40617	V40600 - V40617	V40600 - V40677	V40600 - V40777
Timer Status Bits	V41100 - V41103	V41100 - V41107	V41100 - V41117	V41100 - V41177
Counter Status Bits	V41040 - V41143	V41040 - V41147	V41040 - V41147	V41140 - V41157
Stages	V41000 - V41017	V41000 - V41037	V41000 - V41077	V41000 - V41077

How to Read the Tables

Some of the instructions can have more than one parameter. For example, the SET instruction shown in the ladder program to the right can set a single point or a range of points.

In these cases, execution times depend on the number and type of parameters. The execution time tables list execution times for both situations, as shown below:



SET	1st #: X, Y, C, 2nd #: X, Y, C, S (N pt)	9.2 µs 9.6 µs + 0.9 µs x N
RST	1st #: X, Y, C, 2nd #: X, Y, C, S (N pt)	9.2 µs 9.6 µs + 0.9 µs x N

Execution depends on numbers of locations and types of data used

Boolean Instructions

Boolean Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
STR	X, Y, C, T, CT, S, SP	3.3 µs	3.3 µs	1.4 µs	1.4 µs	0.67 µs	0 µs	0.67 µs	0 µs
STRN	X, Y, C, T, CT, S, SP	3.9 µs	3.9 µs	1.6 µs	1.6 µs	0.67 µs	0 µs	0.67 µs	0 µs
OR	X, Y, C, T, CT, S, SP	2.7 µs	2.7 µs	1.0 µs	1.0 µs	0.51 µs	0.51 µs	0.51 µs	0.51 µs
ORN	X, Y, C, T, CT, S, SP	3.3 µs	3.3 µs	1.4 µs	1.4 µs	0.55 µs	0.55 µs	0.55 µs	0.55 µs
AND	X, Y, C, T, CT, S, SP	2.1 µs	2.1 µs	0.8 µs	0.8 µs	0.42 µs	0.42 µs	0.42 µs	0.42 µs
ANDN	X, Y, C, T, CT, S, SP	2.7 µs	2.7 µs	1.2 µs	1.2 µs	0.51 µs	0.51 µs	0.51 µs	0.51 µs
ANDSTR	None	1.2 µs	1.2 µs	0.7 µs	0.7 µs	0.37 µs	0.37 µs	0.37 µs	0.37 µs
ORSTR	None	1.2 µs	1.2 µs	0.7 µs	0.7 µs	0.37 µs	0.37 µs	0.37 µs	0.37 µs
OUT	X, Y, C	3.4 µs	3.4 µs	7.95 µs	7.65 µs	1.82 µs	1.82 µs	1.82 µs	1.82 µs
OROUT	X, Y, C	8.6 µs	8.6 µs	8.25 µs	8.4 µs	2.09 µs	2.09 µs	2.09 µs	2.09 µs
NOT		-	-	-	-	1.04 µs	1.04 µs	1.04 µs	1.04 µs
SET	1st #: X, Y, C, S, 2nd #: X, Y, C, S (N pt)	17.4 µs 12.0 µs + 5.4 µs x N	6.8 µs 6.8 µs	11.4 µs 11.0 µs + 7.0 µs x N	8.4 µs 8.4 µs	9.2 µs 9.6 µs + 0.9 µs x N	1.0 µs 1.1 µs	9.2 µs 9.6 µs + 0.9 µs x N	1.0 µs 1.1 µs
RST	1st #: X, Y, C, S 2nd #: X, Y, C, S (N pt)	17.7 µs 10.5 µs + 5.2 µs x N	6.8 µs 6.8 µs	11.4 µs 11.0 µs + 7.0 µs x N	8.4 µs 8.4 µs	9.2 µs 9.6 µs + 0.9 µs x N	1.0 µs 1.1 µs	9.2 µs 9.6 µs + 0.9 µs x N	1.0 µs 1.1 µs
	1st #: T, CT 2nd #: T, CT (N pt)	31.6 µs 17 µs + 14.6 µs x N	6.8 µs 6.8 µs	29.0 µs 24.3 µs + 4.7 µs x N	8.4 µs 8.4 µs	25.7 µs 16.8 µs + 2.7 µs x N	1.1 µs 1.4 µs	25.7 µs 16.8 µs + 2.7 µs x N	1.1 µs 1.4 µs
PAUSE	1wd: Y 2wd: Y (N points)	19.0 µs 15 µs + 4 µs x N	19.0 µs 15 µs + 4 µs x N	13.0 µs 11 µs + 3 µs x N	13.0 µs 11 µs + 3 µs x N	5.6 µs 9.2 µs + 0.3 µs x N	5.4 µs 4.8 µs	5.6 µs 9.2 µs + 0.3 µs x N	5.4 µs 4.8 µs

Comparative Boolean Instructions

Comparative Boolean Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
STRE	*1st*	*2nd*								
	V: Data Reg.	V: Data Reg.	77 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	158 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	240 µs	13.8 µs	225 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	139 µs	13.8 µs	135 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	231 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	324 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.7 µs	27.7 µs	27.7 µs	27.7 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.7 µs	27.7 µs	27.7 µs	27.7 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
STRNE	*1st*	*2nd*								
	V: Data Reg.	V: Data Reg.	77 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	13.8 µs	136 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	158 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	240 µs	13.8 µs	225 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	139 µs	13.8 µs	135 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	231 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	324 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		V: Bit Reg.	-	-	-	-	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		V: Bit Reg.	-	-	-	-	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Comparative Boolean Instructions (cont'd)

Comparative Boolean Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
ORE	*1st*	*2nd*								
	V: Data Reg.	V: Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	140 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	239 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	230 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	324 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		V: Bit Reg.	-	-	-	-	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	50.4 µs	50.4 µs	50.4 µs	50.4 µs
		P: Indir. (Bit)	-	-	-	-	50.4 µs	50.4 µs	50.4 µs	50.4 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		V: Bit Reg.	-	-	-	-	30.3 µs	30.3 µs	30.3 µs	30.3 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	50.4 µs	50.4 µs	50.4 µs	50.4 µs
		P: Indir. (Bit)	-	-	-	-	50.4 µs	50.4 µs	50.4 µs	50.4 µs
ORNE	*1st*	*2nd*								
	V: Data Reg.	V: Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	141 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	239 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	230 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	323 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Comparative Boolean Instructions (cont'd)

Comparative Boolean Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
ANDE	*1st*	*2nd*								
	V: Data Reg.	V: Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	139 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	233 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	239 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	229 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	322 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
ANDNE	*1st*	*2nd*								
	V: Data Reg.	V: Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	133 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	139 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	233 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	239 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	229 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	323 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Comparative Boolean Instructions (cont'd)

Comparative Boolean Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
STR	*1st*	*2nd*								
	T, CT	V: Data Reg.	76 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	*2nd*								
	V: Data Reg.	V: Data Reg.	78 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	159 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	159 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	241 µs	13.8 µs	225 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	139 µs	13.8 µs	135 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	231 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	324 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Comparative Boolean Instructions (cont'd)

Comparative Boolean Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
STRN	***1st***	***2nd***								
	T, CT	V: Data Reg.	78 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	13.8 µs	136 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	***2nd***								
	V: Data Reg.	V: Data Reg.	78 µs	13.8 µs	46 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	159 µs	13.8 µs	135 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	57 µs	13.8 µs	46 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	141 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	235 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	159 µs	13.8 µs	136 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	241 µs	13.8 µs	225 µs	16.2 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	139 µs	13.8 µs	135 µs	16.2 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	231 µs	111.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	324 µs	115.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Comparative Boolean Instructions (cont'd)

Comparative Boolean Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
OR	*1st*	*2nd*								
	T, CT	V: Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	140 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	*2nd*								
	V: Data Reg.	V: Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	140 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	240 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	230 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	323 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Comparative Boolean Instructions (cont'd)

Comparative Boolean Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
ORN	***1st***	***2nd***								
	T, CT	V: Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	140 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	***2nd***								
	V: Data Reg.	V: Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	141 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	234 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	240 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	230 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	324 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Comparative Boolean Instructions (cont'd)

Comparative Boolean Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
AND	***1st***	***2nd***								
	T, CT	V: Data Reg.	76 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	139 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	233 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	***2nd***								
	V: Data Reg.	V: Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	140 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	233 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	240 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	229 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	323 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Comparative Boolean Instructions (cont'd)

Comparative Boolean Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
ANDN	***1st***	***2nd***								
	T, CT	V: Data Reg.	76 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	139 µs	110.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	233 µs	114.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	1st	***2nd***								
	V: Data Reg.	V: Data Reg.	75 µs	12.0 µs	44 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	55 µs	12.0 µs	44 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	139 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	233 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	V: Bit Reg.	V: Data Reg.	158 µs	12.0 µs	134 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		V: Bit Reg.	240 µs	12.0 µs	223 µs	13.9 µs	7.6 µs	7.6 µs	7.6 µs	7.6 µs
		K: Constant	137 µs	12.0 µs	133 µs	13.9 µs	4.8 µs	4.8 µs	4.8 µs	4.8 µs
		P: Indir. (Data)	-	-	229 µs	109.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	322 µs	113.0 µs	30.2 µs	30.2 µs	30.2 µs	30.2 µs
	P: Indir. (Data)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
	P: Indir. (Bit)	V: Data Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		V: Bit Reg.	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
		K: Constant	-	-	-	-	27.4 µs	27.4 µs	27.4 µs	27.4 µs
		P: Indir. (Data)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs
		P: Indir. (Bit)	-	-	-	-	51.0 µs	51.0 µs	51.0 µs	51.0 µs

Bit of Word Boolean Instructions

Bit of Word Boolean Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
STRB	V: Data Reg.	-	-	-	-	3.1 µs	3.1 µs	3.1 µs	3.1 µs
	V: Bit Reg.	-	-	-	-	3.1 µs	3.1 µs	3.1 µs	3.1 µs
	P: Indir. (Data)	-	-	-	-	30.0 µs	30.0 µs	30.0 µs	30.0 µs
	P: Indir. (Bit)	-	-	-	-	30.0 µs	30.0 µs	30.0 µs	30.0 µs
STRNB	V: Data Reg.	-	-	-	-	3.0 µs	3.0 µs	3.0 µs	3.0 µs
	V: Bit Reg.	-	-	-	-	3.0 µs	3.0 µs	3.0 µs	3.0 µs
	P: Indir. (Data)	-	-	-	-	29.8 µs	29.8 µs	29.8 µs	29.8 µs
	P: Indir. (Bit)	-	-	-	-	29.8 µs	29.8 µs	29.8 µs	29.8 µs
ORB	V: Data Reg.	-	-	-	-	2.9 µs	2.9 µs	2.9 µs	2.9 µs
	V: Bit Reg.	-	-	-	-	2.9 µs	2.9 µs	2.9 µs	2.9 µs
	P: Indir. (Data)	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
	P: Indir. (Bit)	-	-	-	-	29.9 µs	29.9 µs	29.9 µs	29.9 µs
ORNB	V: Data Reg.	-	-	-	-	2.8 µs	2.8 µs	2.8 µs	2.8 µs
	V: Bit Reg.	-	-	-	-	2.8 µs	2.8 µs	2.8 µs	2.8 µs
	P: Indir. (Data)	-	-	-	-	29.6 µs	29.6 µs	29.6 µs	29.6 µs
	P: Indir. (Bit)	-	-	-	-	29.6 µs	29.6 µs	29.6 µs	29.6 µs
ANDB	V: Data Reg.	-	-	-	-	2.8 µs	2.8 µs	2.8 µs	2.8 µs
	V: Bit Reg.	-	-	-	-	2.8 µs	2.8 µs	2.8 µs	2.8 µs
	P: Indir. (Data)	-	-	-	-	29.6 µs	29.6 µs	29.6 µs	29.6 µs
	P: Indir. (Bit)	-	-	-	-	29.6 µs	29.6 µs	29.6 µs	29.6 µs
ANDNB	V: Data Reg.	-	-	-	-	2.7 µs	2.7 µs	2.7 µs	2.7 µs
	V: Bit Reg.	-	-	-	-	2.7 µs	2.7 µs	2.7 µs	2.7 µs
	P: Indir. (Data)	-	-	-	-	29.6 µs	29.6 µs	29.6 µs	29.6 µs
	P: Indir. (Bit)	-	-	-	-	29.6 µs	29.6 µs	29.6 µs	29.6 µs
OUTB	V: Data Reg.	-	-	-	-	3.1 µs	3.4 µs	3.1 µs	3.4 µs
	V: Bit Reg.	-	-	-	-	3.1 µs	3.4 µs	3.1 µs	3.4 µs
	P: Indir. (Data)	-	-	-	-	30.3 µs	30.7 µs	30.3 µs	30.7 µs
	P: Indir. (Bit)	-	-	-	-	30.3 µs	30.7 µs	30.3 µs	30.7 µs
SETB	V: Data Reg.	-	-	-	-	13.4 µs	3.4 µs	13.4 µs	3.4 µs
	V: Bit Reg.	-	-	-	-	13.4 µs	3.4 µs	13.4 µs	3.4 µs
	P: Indir. (Data)	-	-	-	-	41.1 µs	29.1 µs	41.1 µs	29.1 µs
	P: Indir. (Bit)	-	-	-	-	41.1 µs	29.1 µs	41.1 µs	29.1 µs
RSTB	V: Data Reg.	-	-	-	-	13.5 µs	1.4 µs	13.5 µs	1.4 µs
	V: Bit Reg.	-	-	-	-	13.5 µs	1.4 µs	13.5 µs	1.4 µs
	P: Indir. (Data)	-	-	-	-	41.3 µs	29.1 µs	41.3 µs	29.1 µs
	P: Indir. (Bit)	-	-	-	-	41.3 µs	29.1 µs	41.3 µs	29.1 µs

Immediate Instructions

Immediate Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
LDI	V		-	-	-	-	-	-	20.6 µs	1.1 µs
LDIF	1st#:	X	-	-	-	-	-	-	-	-
	2nd#:	K: Constant	-	-	-	-	-	-	26.6 µs + 0.9 µs x N	1.4 µs
STRI	X		27 µs	9.8 µs	29 µs	10.7 µs	19.3 µs	19.3 µs	19.3 µs	19.3 µs
STRNI	X		26 µs	8.6 µs	29 µs	10.7 µs	19.4 µs	19.4 µs	19.4 µs	19.4 µs
ORI	X		27 µs	9.8 µs	29 µs	8.4 µs	19.1 µs	18.7 µs	19.1 µs	18.7 µs
ORNI	X		26 µs	8.6 µs	29 µs	8.4 µs	19.2 µs	18.9 µs	19.2 µs	18.9 µs
ANDI	X		25 µs	8.0 µs	27 µs	8.4 µs	18.7 µs	18.7 µs	18.7 µs	18.7 µs
ANDNI	X		24 µs	6.8 µs	28 µs	8.4 µs	18.8 µs	18.8 µs	18.8 µs	18.8 µs
OROUTI	Y		45 µs	45 µs	39 µs	40 µs	27.5 µs	27.5 µs	27.5 µs	27.5 µs
OUTI	Y		45 µs	45 µs	39 µs	40 µs	25.5 µs	25.5 µs	25.5 µs	25.5 µs
OUTIF	1st#:	Y	-	-	-	-	-	-	-	-
	2nd#:	K: Constant	-	-	-	-	-	-	66.1 µs + 0.9 µs x N	1.4 µs
SETI	1st#:	Y	25.5 µs	6.8 µs	39.0 µs	8.4 µs	23.1 µs	0.9 µs	23.1 µs	0.9 µs
	2nd#:	Y (N pt)	5.5 µs + 20 µs x N	6.8 µs	44 µs + 25 µs x N	8.4 µs	22.8 µs + 1.4 µs x N	0.9 µs	22.8 µs + 1.4 µs x N	0.9 µs
RSTI	1st#:	Y	25.5 µs	6.8 µs	37 µs	8.4 µs	23.2 µs	0.9 µs	23.2 µs	0.9 µs
	2nd#:	Y (N pt)	5 µs + 20.5 µs x N	6.8 µs	45 µs + 22 µs x N	8.4 µs	22.8 µs + 1.4 µs x N	0.9 µs	22.8 µs + 1.4 µs x N	0.9 µs

Timer, Counter and Shift Register Instructions

Timer, Counter and Shift Register Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
	1st	*2nd*								
TMR	T	V: Data Reg.	75 µs	31 µs	61 µs	23.5 µs	26.8 µs	7.3 µs	26.8 µs	7.3 µs
		V: Bit Reg.	158 µs	31 µs	158 µs	23.5 µs	26.8 µs	7.3 µs	26.8 µs	7.3 µs
		K: Constant	66 µs	31 µs	70 µs	23.5 µs	20.0 µs	4.8 µs	20.0 µs	4.8 µs
		P: Indir. (Data)	-	-	177 µs	131.0 µs	45.6 µs	30.2 µs	45.6 µs	30.2 µs
		P: Indir. (Bit)	-	-	271 µs	136.0 µs	45.6 µs	30.2 µs	45.6 µs	30.2 µs
TMRF	T	V: Data Reg.	75 µs	31 µs	61 µs	23.5 µs	51.4 µs	7.3 µs	51.4 µs	7.3 µs
		V: Bit Reg.	158 µs	31 µs	158 µs	23.5 µs	51.4 µs	7.3 µs	51.4 µs	7.3 µs
		K: Constant	66 µs	31 µs	70 µs	23.5 µs	48.4 µs	4.6 µs	48.4 µs	4.6 µs
		P: Indir. (Data)	-	-	177 µs	131.0 µs	75.9 µs	30.2 µs	75.9 µs	30.2 µs
		P: Indir. (Bit)	-	-	271 µs	136.0 µs	75.9 µs	30.2 µs	75.9 µs	30.2 µs
TMRA	T	V: Data Reg.	94 µs	56 µs	75 µs	41 µs	48.9 µs	7.3 µs	48.9 µs	7.3 µs
		V: Bit Reg.	304 µs	264 µs	253 µs	219 µs	48.9 µs	7.3 µs	48.9 µs	7.3 µs
		K: Constant	95 µs	45 µs	79 µs	49 µs	45.0 µs	4.6 µs	45.0 µs	4.6 µs
		P: Indir. (Data)	-	-	193 µs	159 µs	75.9 µs	30.2 µs	75.9 µs	30.2 µs
		P: Indir. (Bit)	-	-	366 µs	331 µs	75.9 µs	30.2 µs	75.9 µs	30.2 µs
TMRAF	T	V: Data Reg.	98 µs	54 µs	75 µs	42 µs	54.2 µs	7.3 µs	54.2 µs	7.3 µs
		V: Bit Reg.	304 µs	264 µs	253 µs	218 µs	54.2 µs	7.3 µs	54.2 µs	7.3 µs
		K: Constant	95 µs	49 µs	80 µs	50 µs	50.3 µs	4.6 µs	50.3 µs	4.6 µs
		P: Indir. (Data)	-	-	193 µs	159 µs	81.2 µs	30.2 µs	81.2 µs	30.2 µs
		P: Indir. (Bit)	-	-	366 µs	331 µs	81.2 µs	30.2 µs	81.2 µs	30.2 µs
CNT	CT	V: Data Reg.	68 µs	61 µs	59 µs	38 µs	25.8 µs	7.3 µs	25.8 µs	7.3 µs
		V: Bit Reg.	148 µs	141 µs	157 µs	133 µs	25.8 µs	7.3 µs	25.8 µs	7.3 µs
		K: Constant	56 µs	45 µs	59 µs	45 µs	22.2 µs	4.6 µs	22.2 µs	4.6 µs
		P: Indir. (Data)	-	-	176 µs	152 µs	53.5 µs	30.2 µs	53.5 µs	30.2 µs
		P: Indir. (Bit)	-	-	270 µs	245 µs	53.5 µs	30.2 µs	53.5 µs	30.2 µs
SGCNT	CT	V: Data Reg.	57 µs	64 µs	58 µs	38 µs	27.3 µs	7.3 µs	27.3 µs	7.3 µs
		V: Bit Reg.	140 µs	148 µs	155 µs	133 µs	27.3 µs	7.3 µs	27.3 µs	7.3 µs
		K: Constant	46 µs	53 µs	67 µs	45 µs	23.5 µs	4.6 µs	23.5 µs	4.6 µs
		P: Indir. (Data)	-	-	175 µs	152 µs	54.9 µs	30.2 µs	54.9 µs	30.2 µs
		P: Indir. (Bit)	-	-	268 µs	245 µs	54.9 µs	30.2 µs	54.9 µs	30.2 µs
UDC	CT	V: Data Reg.	103 µs	74 µs	80 µs	56 µs	39.8 µs	7.3 µs	39.8 µs	7.3 µs
		V: Bit Reg.	310 µs	281 µs	261 µs	224 µs	39.8 µs	7.3 µs	39.8 µs	7.3 µs
		K: Constant	102 µs	70 µs	97 µs	60 µs	35.4 µs	4.6 µs	35.4 µs	4.6 µs
		P: Indir. (Data)	-	-	202 µs	165 µs	67.8 µs	30.2 µs	67.8 µs	30.2 µs
		P: Indir. (Bit)	-	-	374 µs	336 µs	67.8 µs	30.2 µs	67.8 µs	30.2 µs
SR	C (N points to shift)		30 µs + 4.6 µs x N	17.2 µs	25 µs + 4 µs x N	19.7 µs	17.8 µs + 0.9 µs x N	9.8 µs	17.8 µs + 0.9 µs x N	9.8 µs

Accumulator Data Instructions

Accumulator/Stack Load and Output Data Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	**Legal Data Types**		**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
LD	V: Data Reg.		68 µs	8.4 µs	68 µs	8.4 µs	11.8 µs	1.0 µs	11.8 µs	1.0 µs
	V: Bit Reg.		149 µs	8.4 µs	143 µs	8.4 µs	11.8 µs	1.0 µs	11.8 µs	1.0 µs
	K: Constant		62 µs	8.4 µs	159 µs	8.4 µs	9.0 µs	1.0 µs	9.0 µs	1.0 µs
	P: Indir. (Data)		169 µs	8.4 µs	238 µs	8.4 µs	33.9 µs	0.9 µs	33.9 µs	0.9 µs
	P: Indir. (Bit)		256 µs	8.4 µs	62 µs	8.4 µs	33.9 µs	0.9 µs	33.9 µs	0.9 µs
LDA	O: (Octal constant for address)		58 µs	8.4 µs	56 µs	8.4 µs	10.4 µs	1.0 µs	10.4 µs	1.0 µs
LDD	V: Data Reg.		72 µs	8.4 µs	67 µs	8.4 µs	12.2 µs	1.0 µs	12.2 µs	1.0 µs
	V: Bit Reg.		266 µs	8.4 µs	228 µs	8.4 µs	12.2 µs	1.0 µs	12.2 µs	1.0 µs
	K: Constant		64 µs	8.4 µs	69 µs	8.4 µs	9.0 µs	1.0 µs	9.0 µs	1.0 µs
	P: Indir. (Data)		172 µs	8.4 µs	158 µs	8.4 µs	37.8 µs	0.9 µs	37.8 µs	0.9 µs
	P: Indir. (Bit)		373 µs	8.4 µs	323 µs	8.4 µs	37.8 µs	0.9 µs	37.8 µs	0.9 µs
LDF	**1st**	**2nd**								
	X, Y, C, S, T, CT, SP	K: Constant	-	-	86 µs + 5 µs x N	8.4 µs	20.5 µs + 0.9 µs x N	0.9 µs	20.5 µs + 0.9 µs x N	0.9 µs
LDR	V: Data Reg.		-	-	-	-	29.5 µs	1.0 µs	29.5 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	29.5 µs	1.0 µs	29.5 µs	1.0 µs
	K: Constant		-	-	-	-	25.5 µs	1.0 µs	25.5 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	54.9 µs	1.0 µs	54.9 µs	1.0 µs
	P: Indir. (Bit)		-	-	-	-	54.9 µs	1.0 µs	54.9 µs	1.0 µs
LDSX	K: Constant		-	-	79 µs	8.4 µs	14.6 µs	1.0 µs	14.6 µs	1.0 µs
LDX	V: Data Reg.		-	-	-	-	10.8 µs	1.0 µs	10.8 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	10.8 µs	1.0 µs	10.8 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	45.2 µs	1.0 µs	45.2 µs	1.0 µs
	P: Indir. (Bit)		-	-	-	-	45.2 µs	1.0 µs	45.2 µs	1.0 µs
OUT	V: Data Reg.		60 µs	8.4 µs	21 µs	8.4 µs	9.3 µs	1.0 µs	9.3 µs	1.0 µs
	V: Bit Reg.		132 µs	8.4 µs	126 µs	8.4 µs	9.3 µs	1.0 µs	9.3 µs	1.0 µs
	P: Indir. (Data)		162 µs	8.4 µs	112 µs	8.4 µs	35.2 µs	0.9 µs	35.2 µs	0.9 µs
	P: Indir. (Bit)		239 µs	8.4 µs	222 µs	8.4 µs	35.2 µs	0.9 µs	35.2 µs	0.9 µs
OUTD	V: Data Reg.		68 µs	8.4 µs	26 µs	8.4 µs	10.2 µs	1.0 µs	10.2 µs	1.0 µs
	V: Bit Reg.		276 µs	8.4 µs	235 µs	8.4 µs	10.2 µs	1.0 µs	10.2 µs	1.0 µs
	P: Indir. (Data)		196 µs	8.4 µs	116 µs	8.4 µs	35.8 µs	0.9 µs	35.8 µs	0.9 µs
	P: Indir. (Bit)		384 µs	8.4 µs	331 µs	8.4 µs	35.8 µs	0.9 µs	35.8 µs	0.9 µs

Accumulator Data Instructions (cont'd)

Accumulator/Stack Load and Output Data Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	**Legal Data Types**		**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
OUTF	**1st**	**2nd**								
	X, Y, C	K: Constant	-	-	53 µs + 7 µs x N	8.4 µs	54 µs + 1.0 µs x N	0.9 µs	54 µs + 1.0 µs x N	0.9 µs
OUTL	V: Data Reg.		-	-	-	-	-	-	13.5 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	-	-	13.5 µs	1.0 µs
OUTM	V: Data Reg.		-	-	-	-	-	-	13.7 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	-	-	13.7 µs	1.0 µs
OUTX	V: Data Reg.		-	-	-	-	-	-	17.2 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	-	-	17.2 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	-	-	43.4 µs	1.0 µs
	P: Indir. (Bit)		-	-	-	-	-	-	43.4 µs	1.0 µs
POP	None		55 µs	7.2 µs	50 µs	8.4 µs	8.4 µs	1.0 µs	8.4 µs	1.0 µs

Logical Instructions

Logical (Accumulator) Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
AND	V: Data Reg.		58 µs	10.4 µs	54 µs	8.4 µs	7.9 µs	1.0 µs	7.9 µs	1.0 µs
	V: Bit Reg.		261 µs	10.4 µs	145 µs	8.4 µs	7.9 µs	1.0 µs	7.9 µs	1.0 µs
	P: Indir. (Data)		-	-	162 µs	8.4 µs	33.4 µs	0.9 µs	33.4 µs	0.9 µs
	P: Indir. (Bit)		-	-	241 µs	8.4 µs	33.4 µs	0.9 µs	33.4 µs	0.9 µs
ANDD	V: Data Reg.		-	-	-	-	8.9 µs	1.0 µs	8.9 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	8.9 µs	1.0 µs	8.9 µs	1.0 µs
	K: Constant		53 µs	8.4 µs	60 µs	8.4 µs	5.7 µs	1.0 µs	5.7 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	34.4 µs	0.9 µs	34.4 µs	0.9 µs
	P: Indir. (Bit)		-	-	-	-	34.4 µs	0.9 µs	34.4 µs	0.9 µs
ANDF	**1st**	**2nd**								
	X, Y, C, S, T, CT, SP GX, GY	K: Constant	-	-	-	-	21.6 µs + 0.9 µs x N	1.0 µs	21.6 µs + 0.9 µs x N	1.0 µs
ANDS	None		-	-	-	-	-	-	10.0 µs	1.0 µs
OR	V: Data Reg.		59 µs	10.4 µs	54 µs	8.4 µs	8.1 µs	1.0 µs	8.1 µs	1.0 µs
	V: Bit Reg.		257 µs	10.4 µs	144 µs	8.4 µs	8.1 µs	1.0 µs	8.1 µs	1.0 µs
	P: Indir. (Data)		-	-	160 µs	8.4 µs	33.8 µs	0.9 µs	33.8 µs	0.9 µs
	P: Indir. (Bit)		-	-	239 µs	8.4 µs	33.8 µs	0.9 µs	33.8 µs	0.9 µs
ORD	V: Data Reg.		-	-	-	-	9.0 µs	1.0 µs	9.0 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	9.0 µs	1.0 µs	9.0 µs	1.0 µs
	K: Constant		49 µs	8.4 µs	60 µs	8.4 µs	5.8 µs	1.0 µs	5.8 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	34.5 µs	0.9 µs	34.5 µs	0.9 µs
	P: Indir. (Bit)		-	-	-	-	34.5 µs	0.9 µs	34.5 µs	0.9 µs
ORF	**1st**	**2nd**								
	X, Y, C, S, T, CT, SP GX, GY	K: Constant	-	-	-	-	20.9 µs + 0.9 µs x N	1.0 µs	20.9 µs + 0.9 µs x N	1.0 µs
ORS	None		-	-	-	-	-	-	10.2 µs	1.0 µs
XOR	V: Data Reg.		60 µs	10.4 µs	69 µs	8.4 µs	8.0 µs	1.0 µs	8.0 µs	1.0 µs
	V: Bit Reg.		257 µs	10.4 µs	144 µs	8.4 µs	8.0 µs	1.0 µs	8.0 µs	1.0 µs
	P: Indir. (Data)		-	-	160 µs	8.4 µs	33.6 µs	0.9 µs	33.6 µs	0.9 µs
	P: Indir. (Bit)		-	-	239 µs	8.4 µs	33.6 µs	0.9 µs	33.6 µs	0.9 µs
XORD	V: Data Reg.		-	-	-	-	9.0 µs	1.0 µs	9.0 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	9.0 µs	1.0 µs	9.0 µs	1.0 µs
	K: Constant		49 µs	8.4 µs	62 µs	8.4 µs	5.4 µs	1.0 µs	5.4 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	34.4 µs	0.9 µs	34.4 µs	0.9 µs
	P: Indir. (Bit)		-	-	-	-	34.4 µs	0.9 µs	34.4 µs	0.9 µs

Logical Instructions (cont'd)

Logical (Accumulator) Instructions			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	**Legal Data Types**		**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
XORF	**1st**	**2nd**								
	X, Y, C, S, T, CT, SP GX, GY	K: Constant	-	-	-	-	20.9 µs + 0.9 µs x N	1.0 µs	20.9 µs + 0.9 µs x N	1.0 µs
XORS	None		-	-	-	-	-	-	10.1 µs	1.0 µs
CMP	V: Data Reg.		59 µs	10.4 µs	69 µs	8.4 µs	9.4 µs	1.0 µs	9.4 µs	1.0 µs
	V: Bit Reg.		259 µs	10.4 µs	115 µs	8.4 µs	9.4 µs	1.0 µs	9.4 µs	1.0 µs
	P: Indir. (Data)		-	-	130 µs	8.4 µs	34.9 µs	0.9 µs	34.9 µs	0.9 µs
	P: Indir. (Bit)		-	-	211 µs	8.4 µs	34.9 µs	0.9 µs	34.9 µs	0.9 µs
CMPD	V: Data Reg.		63 µs	8.4 µs	47 µs	8.4 µs	9.9 µs	1.0 µs	9.9 µs	1.0 µs
	V: Bit Reg.		257 µs	8.4 µs	206 µs	8.4 µs	9.9 µs	1.0 µs	9.9 µs	1.0 µs
	K: Constant		54 µs	8.4 µs	49 µs	8.4 µs	6.7 µs	1.0 µs	6.7 µs	1.0 µs
	P: Indir. (Data)		-	-	133 µs	8.4 µs	35.4 µs	1.0 µs	35.4 µs	1.0 µs
	P: Indir. (Bit)		-	-	303 µs	8.4 µs	35.4 µs	1.0 µs	35.4 µs	1.0 µs
CMPF	**1st**	**2nd**								
	X, Y, C, S, T, CT, SP GX, GY	K: Constant	-	-	-	-	29.2 µs+ 1.0 µs x N	1.0 µs	29.2 µs+ 1.0 µs x N	1.0 µs
CMPR	V: Data Reg.		-	-	-	-	42.8 µs	1.0 µs	42.8 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	42.8 µs	1.0 µs	42.8 µs	1.0 µs
	K: Constant		-	-	-	-	38.4 µs	1.0 µs	38.4 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	69.0 µs	1.0 µs	69.0 µs	1.0 µs
	P: Indir. (Bit)		-	-	-	-	69.0 µs	1.0 µs	69.0 µs	1.0 µs
CMPS	None		-	-	-	-	-	-	11.2 µs	1.0 µs

Math Instructions

Math Instructions (Accumulator)		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
ADD	V: Data Reg.	198 µs	10.6 µs	291 µs	8.4 µs	78.4 µs	0.9 µs	78.4 µs	0.9 µs
	V: Bit Reg.	397 µs	10.6 µs	363 µs	8.4 µs	78.4 µs	0.9 µs	78.4 µs	0.9 µs
	P: Indir. (Data)	-	-	441 µs	8.4 µs	101.2 µs	0.9 µs	101.2 µs	0.9 µs
	P: Indir. (Bit)	-	-	520 µs	8.4 µs	101.2 µs	0.9 µs	101.2 µs	0.9 µs
ADDD	V: Data Reg.	198 µs	8.4 µs	291 µs	8.4 µs	83.3 µs	0.9 µs	83.3 µs	0.9 µs
	V: Bit Reg.	397 µs	8.4 µs	512 µs	8.4 µs	83.3 µs	0.9 µs	83.3 µs	0.9 µs
	K: Constant	188 µs	8.4 µs	298 µs	8.4 µs	67.7 µs	0.9 µs	67.7 µs	0.9 µs
	P: Indir. (Data)	-	-	442 µs	8.4 µs	101.2 µs	0.9 µs	101.2 µs	0.9 µs
	P: Indir. (Bit)	-	-	608 µs	8.4 µs	101.2 µs	0.9 µs	101.2 µs	0.9 µs
SUB	V: Data Reg.	200 µs	10.6 µs	287 µs	8.4 µs	77.4 µs	0.9 µs	77.4 µs	0.9 µs
	V: Bit Reg.	397 µs	10.6 µs	360 µs	8.4 µs	77.4 µs	0.9 µs	77.4 µs	0.9 µs
	P: Indir. (Data)	-	-	434 µs	8.4 µs	95.1 µs	0.9 µs	95.1 µs	0.9 µs
	P: Indir. (Bit)	-	-	513 µs	8.4 µs	95.1 µs	0.9 µs	95.1 µs	0.9 µs
SUBD	V: Data Reg.	198 µs	8.4 µs	288 µs	8.4 µs	82.5 µs	0.9 µs	82.5 µs	0.9 µs
	V: Bit Reg.	392 µs	8.4 µs	504 µs	8.4 µs	82.5 µs	0.9 µs	82.5 µs	0.9 µs
	K: Constant	190 µs	8.4 µs	294 µs	8.4 µs	66.0 µs	0.9 µs	66.0 µs	0.9 µs
	P: Indir. (Data)	-	-	434 µs	8.4 µs	99.7 µs	0.9 µs	99.7 µs	0.9 µs
	P: Indir. (Bit)	-	-	600 µs	8.4 µs	99.7 µs	0.9 µs	99.7 µs	0.9 µs
MUL	V: Data Reg.	497 µs	10.6 µs	311 µs	8.4 µs	266.1 µs	0.9 µs	266.1 µs	0.9 µs
	V: Bit Reg.	483 µs	10.6 µs	385 µs	8.4 µs	266.1 µs	0.9 µs	266.1 µs	0.9 µs
	K: Constant	487 µs	8.4 µs	334 µs	8.4 µs	286.9 µs	0.9 µs	286.9 µs	0.9 µs
	P: Indir. (Data)	-	-	401 µs	8.4 µs	290.0 µs	0.9 µs	290.0 µs	0.9 µs
	P: Indir. (Bit)	-	-	461 µs	8.4 µs	290.0 µs	0.9 µs	290.0 µs	0.9 µs
MULD	V: Data Reg.					839.1 µs	0.9 µs	839.1 µs	0.9 µs
	V: Bit Reg.	-	-	-	-	839.1 µs	0.9 µs	839.1 µs	0.9 µs
	P: Indir. (Data)	-	-	-	-	863.1 µs	0.9 µs	863.1 µs	0.9 µs
	P: Indir. (Bit)					863.1 µs	0.9 µs	863.1 µs	0.9 µs
DIV	V: Data Reg.	909 µs	10.6 µs	601 µs	8.4 µs	363.9 µs	0.9 µs	363.9 µs	0.9 µs
	V: Bit Reg.	1108 µs	10.6 µs	675 µs	8.4 µs	363.9 µs	0.9 µs	363.9 µs	0.9 µs
	K: Constant	699 µs	8.4 µs	573 µs	8.4 µs	384.4 µs	0.9 µs	384.4 µs	0.9 µs
	P: Indir. (Data)	-	-	691 µs	8.4 µs	419.8 µs	0.9 µs	419.8 µs	0.9 µs
	P: Indir. (Bit)	-	-	771 µs	8.4 µs	419.8 µs	0.9 µs	419.8 µs	0.9 µs
DIVD	V: Data Reg.					398.3 µs	0.9 µs	398.3 µs	0.9 µs
	V: Bit Reg.	-	-	-	-	398.3 µs	0.9 µs	398.3 µs	0.9 µs
	P: Indir. (Data)	-	-	-	-	390.9 µs	0.9 µs	390.9 µs	0.9 µs
	P: Indir. (Bit)					390.9 µs	0.9 µs	390.9 µs	0.9 µs
INC	V: Data Reg.					48.5 µs	1.0 µs	48.5 µs	1.0 µs
	V: Bit Reg.	-	-	-	-	48.5 µs	1.0 µs	48.5 µs	1.0 µs
	P: Indir. (Data)	-	-	-	-	74.7 µs	1.0 µs	74.7 µs	1.0 µs
	P: Indir. (Bit)					74.7 µs	1.0 µs	74.7 µs	1.0 µs

Math Instructions (cont'd)

Math Instructions (Accumulator)		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
DEC	V: Data Reg.	-	-	-	-	47.5 µs	1.0 µs	47.5 µs	1.0 µs
	V: Bit Reg.	-	-	-	-	47.5 µs	1.0 µs	47.5 µs	1.0 µs
	P: Indir. (Data)	-	-	-	-	71.5 µs	1.0 µs	71.5 µs	1.0 µs
	P: Indir. (Bit)	-	-	-	-	71.5 µs	1.0 µs	71.5 µs	1.0 µs
INCB	V: Data Reg.	88 µs	10.4 µs	35 µs	8.4 µs	13.2 µs	1.0 µs	13.2 µs	1.0 µs
	V: Bit Reg.	349 µs	10.4 µs	211 µs	8.4 µs	13.2 µs	1.0 µs	13.2 µs	1.0 µs
	P: Indir. (Data)	-	-	126 µs	8.4 µs	38.6 µs	0.9 µs	38.6 µs	0.9 µs
	P: Indir. (Bit)	-	-	307 µs	8.4 µs	38.6 µs	0.9 µs	38.6 µs	0.9 µs
DECB	V: Data Reg.	82 µs	10.4 µs	33 µs	8.4 µs	13.2 µs	1.0 µs	13.2 µs	1.0 µs
	V: Bit Reg.	351 µs	10.4 µs	210 µs	8.4 µs	13.2 µs	1.0 µs	13.2 µs	1.0 µs
	P: Indir. (Data)	-	-	123 µs	8.4 µs	38.0 µs	0.9 µs	38.0 µs	0.9 µs
	P: Indir. (Bit)	-	-	304 µs	8.4 µs	38.0 µs	0.9 µs	38.0 µs	0.9 µs
ADDB	V: Data Reg.	-	-	-	-	24.9 µs	1.0 µs	24.9 µs	1.0 µs
	V: Bit Reg.	-	-	-	-	24.9 µs	1.0 µs	24.9 µs	1.0 µs
	K: Constant	-	-	-	-	23.5 µs	1.0 µs	23.5 µs	1.0 µs
	P: Indir. (Data)	-	-	-	-	51.1 µs	1.0 µs	51.1 µs	1.0 µs
	P: Indir. (Bit)	-	-	-	-	51.1 µs	1.0 µs	51.1 µs	1.0 µs
ADDBD	V: Data Reg.	-	-	-	-	-	-	24.4 µs	1.0 µs
	V: Bit Reg.	-	-	-	-	-	-	24.4 µs	1.0 µs
	K: Constant	-	-	-	-	-	-	20.7 µs	1.0 µs
	P: Indir. (Data)	-	-	-	-	-	-	50.7 µs	1.0 µs
	P: Indir. (Bit)	-	-	-	-	-	-	50.7 µs	1.0 µs
SUBB	V: Data Reg.	-	-	-	-	24.7 µs	1.0 µs	24.7 µs	1.0 µs
	V: Bit Reg.	-	-	-	-	24.7 µs	1.0 µs	24.7 µs	1.0 µs
	K: Constant	-	-	-	-	23.3 µs	1.0 µs	23.3 µs	1.0 µs
	P: Indir. (Data)	-	-	-	-	50.6 µs	1.0 µs	50.6 µs	1.0 µs
	P: Indir. (Bit)	-	-	-	-	50.6 µs	1.0 µs	50.6 µs	1.0 µs
SUBBD	V: Data Reg.	-	-	-	-	-	-	24.2 µs	1.0 µs
	V: Bit Reg.	-	-	-	-	-	-	24.2 µs	1.0 µs
	K: Constant	-	-	-	-	-	-	20.2 µs	1.0 µs
	P: Indir. (Data)	-	-	-	-	-	-	50.2 µs	1.0 µs
	P: Indir. (Bit)	-	-	-	-	-	-	50.2 µs	1.0 µs
MULB	V: Data Reg.	-	-	-	-	10.8 µs	1.0 µs	10.8 µs	1.0 µs
	V: Bit Reg.	-	-	-	-	10.8 µs	1.0 µs	10.8 µs	1.0 µs
	K: Constant	-	-	-	-	8.2 µs	1.0 µs	8.2 µs	1.0 µs
	P: Indir. (Data)	-	-	-	-	37.1 µs	1.0 µs	37.1 µs	1.0 µs
	P: Indir. (Bit)	-	-	-	-	37.1 µs	1.0 µs	37.1 µs	1.0 µs

Math Instructions (cont'd)

Math Instructions (Accumulator)			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
DIVB	V: Data Reg.		-	-	-	-	28.7 µs	1.0 µs	28.7 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	28.7 µs	1.0 µs	28.7 µs	1.0 µs
	K: Constant		-	-	-	-	26.1 µs	1.0 µs	26.1 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	54.9 µs	1.0 µs	54.9 µs	1.0 µs
	P: Indir. (Bit)		-	-	-	-	54.9 µs	1.0 µs	54.9 µs	1.0 µs
ADDR	V: Data Reg.		-	-	-	-	48.1 µs	1.0 µs	48.1 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	48.1 µs	1.0 µs	48.1 µs	1.0 µs
	K: Constant		-	-	-	-	41.7 µs	1.0 µs	41.7 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	74.3 µs	1.0 µs	74.3 µs	1.0 µs
	P: Indir. (Bit)		-	-	-	-	74.3 µs	1.0 µs	74.3 µs	1.0 µs
SUBR	V: Data Reg.		-	-	-	-	50.1 µs	1.0 µs	50.1 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	50.1 µs	1.0 µs	50.1 µs	1.0 µs
	K: Constant		-	-	-	-	58.7 µs	1.0 µs	58.7 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	76.3 µs	1.0 µs	76.3 µs	1.0 µs
	P: Indir. (Bit)		-	-	-	-	76.3 µs	1.0 µs	76.3 µs	1.0 µs
MULR	V: Data Reg.		-	-	-	-	54.2 µs	1.0 µs	54.2 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	54.2 µs	1.0 µs	54.2 µs	1.0 µs
	K: Constant		-	-	-	-	42.7 µs	1.0 µs	42.7 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	80.4 µs	1.0 µs	80.4 µs	1.0 µs
	P: Indir. (Bit)		-	-	-	-	80.4 µs	1.0 µs	80.4 µs	1.0 µs
DIVR	V: Data Reg.		-	-	-	-	50.1 µs	1.0 µs	50.1 µs	1.0 µs
	V: Bit Reg.		-	-	-	-	50.1 µs	1.0 µs	50.1 µs	1.0 µs
	K: Constant		-	-	-	-	58.7 µs	1.0 µs	58.7 µs	1.0 µs
	P: Indir. (Data)		-	-	-	-	76.3 µs	1.0 µs	76.3 µs	1.0 µs
	P: Indir. (Bit)		-	-	-	-	76.3 µs	1.0 µs	76.3 µs	1.0 µs
ADDF	**1st**	**2nd**								
	X, Y, C, S, T, CT, SP GX, GY	K: Constant	-	-	-	-	-	-	109.3 µs + 0.9 µs x N	1.0 µs
SUBF	**1st**	**2nd**								
	X, Y, C, S, T, CT, SP GX, GY	K: Constant	-	-	-	-	-	-	107.3 µs + 0.9 µs x N	1.0 µs
MULF	**1st**	**2nd**								
	X, Y, C, S, T, CT, SP GX, GY	K: Constant	-	-	-	-	-	-	352.5 µs + 0.8 µs x N	1.0 µs

Math Instructions (cont'd)

Math Instructions Accumulator			D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
DIVF	1st	2nd								
	X, Y, C, S, T, CT, SP GX, GY	K: Constant	-	-	-	-	-	-	477.3 µs + 0.8 µs x N	1.0 µs
ADDS	None		-	-	-	-	-	-	99.5 µs	1.0 µs
SUBS	None		-	-	-	-	-	-	97.5 µs	1.0 µs
MULS	None		-	-	-	-	-	-	342.5 µs	1.0 µs
DIVS	None		-	-	-	-	-	-	467.3 µs	1.0 µs
ADDBS	None		-	-	-	-	-	-	24.3 µs	1.0 µs
SUBBS	None		-	-	-	-	-	-	23.7 µs	1.0 µs
MULBS	None		-	-	-	-	-	-	11.7 µs	1.0 µs
DIVBS	None		-	-	-	-	-	-	29.7 µs	1.0 µs
SQRTR	None		-	-	-	-	-	-	87.9 µs	1.0 µs
SINR	None		-	-	-	-	-	-	226.8 µs	1.0 µs
COSR	None		-	-	-	-	-	-	213.1 µs	1.0 µs
TANR	None		-	-	-	-	-	-	285.5 µs	1.0 µs
ASINR	None		-	-	-	-	-	-	489.8 µs	1.0 µs
ACOSR	None		-	-	-	-	-	-	508.3 µs	1.0 µs
ATANR	None		-	-	-	-	-	-	317.1 µs	1.0 µs

Differential Instructions

Differential Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
PD	X, Y, C	13.5 µs	13.5 µs	15.9 µs	14.6 µs	14.4 µs	14.4 µs	14.4 µs	14.4 µs
STRPD	X, Y, C, S, T, CT	-	-	-	-	5.4 µs	5.4 µs	5.4 µs	5.4 µs
STRND	X, Y, C, S, T, CT	-	-	-	-	7.3 µs	7.3 µs	7.3 µs	7.3 µs
ORPD	X, Y, C, S, T, CT	-	-	-	-	6.8 µs	5.2 µs	6.8 µs	5.2 µs
ORND	X, Y, C, S, T, CT	-	-	-	-	7.1 µs	4.9 µs	7.1 µs	4.9 µs
ANDPD	X, Y, C, S, T, CT	-	-	-	-	6.8 µs	5.2 µs	6.8 µs	5.2 µs
ANDND	X, Y, C, S, T, CT	-	-	-	-	7.1 µs	4.9 µs	7.1 µs	4.9 µs

Bit Instructions

Bit Instructions (Accumulator)		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
SUM	None	-	-	-	-	-	-	6.7 µs	1.0 µs
SHFR	V: Data Reg. (N bits)	44 µs + 14 .6 µs x N	10.4 µs	35 µs + 6 µs x N	8.4 µs	12.1 µs + 0.1 µs x N	0.9 µs	12.1 µs + 0.1 µs x N	0.9 µs
	V: Bit Reg (N bits)	243 µs + 14.6 µs x N	8.4 µs	110 µs + 6 µs x N	8.4 µs	12.1 µs + 0.1 µs x N	0.9 µs	12.1 µs + 0.1 µs x N	0.9 µs
	K: Constant (N bits)	34 µs + 14.6 µs x N	8.4 µs	35 µs + 6 µs x N	8.4 µs	8.4 µs + 0.1 µs x N	0.9 µs	8.4 µs + 0.1 µs x N	0.9 µs
SHFL	V: Data Reg. (N bits)	44 µs + 14.6 µs x N	10.4 µs	33 µs + 6 µs x N	8.4 µs	12.1 µs + 0.1 µs x N	0.9 µs	12.1 µs + 0.1 µs x N	0.9 µs
	V: Bit Reg (N bits)	243 µs + 14.6 µs x N	8.4 µs	107 µs + 6 µs x N	8.4 µs	12.1 µs + 0.1 x N	0.9 µs	12.1 µs + 0.1 µs x N	0.9 µs
	K: Constant (N bits)	34 µs + 14.6 µs x N	8.4 µs	33 µs + 6 µs x N	8.4 µs	8.4 µs + 0.1 µs x N	0.9 µs	8.4 µs + 0.1 µs x N	0.9 µs
ROTR	V: Data Reg. (N bits)	-	-	-	-	-	-	16.4 µs	1.0 µs
	V: Bit Reg (N bits)	-	-	-	-	-	-	16.4 µs	1.0 µs
	K: Constant (N bits)	-	-	-	-	-	-	12.9 µs	1.0 µs
ROTL	V: Data Reg. (N bits)	-	-	-	-	-	-	16.4 µs	1.0 µs
	V: Bit Reg (N bits)	-	-	-	-	-	-	16.4 µs	1.0 µs
	K: Constant (N bits)	-	-	-	-	-	-	12.7 µs	1.0 µs
ENCO	None	62 µs	7.2 µs	98 µs	8.4 µs	33.9 µs	0.9 µs	33.9 µs	0.9 µs
DECO	None	34 µs	7.2 µs	28 µs	8.4 µs	5.7 µs	1.0 µs	5.7 µs	1.0 µs

Number Conversion Instructions

Number Conversion Instructions (Accumulator)		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
BIN	None	359 µs	7.2 µs	267 µs	8.4 µs	100.2 µs	0.9 µs	100.2 µs	0.9 µs
BCD	None	403 µs	7.2 µs	383 µs	8.4 µs	95.2 µs	0.9 µs	95.2 µs	0.9 µs
INV	None	27 µs	5.0 µs	12 µs	8.4 µs	2.5 µs	1.0 µs	2.5 µs	1.0 µs
BCDPL	None	296 µs	7.2 µs	69 µs	8.4 µs	75.6 µs	1.0 µs	75.6 µs	1.0 µs
ATH	V	-	-	-	-	-	-	25.4 µs	1.0 µs
HTA	V	-	-	-	-	-	-	25.4 µs	1.0 µs
GRAY	None	-	-	227 µs	9.0 µs	110.8 µs	1.0 µs	110.8 µs	1.0 µs
SFLDGT	None	-	-	258 µs	9.0 µs	23.1 µs	1.0 µs	23.1 µs	1.0 µs
BTOR	None	-	-	-	-	18.6 µs	1.0 µs	18.6 µs	1.0 µs
RTOB	None	-	-	-	-	8.6 µs	1.0 µs	8.6 µs	1.0 µs
RADR	None	-	-	-	-	-	-	51.4 µs	1.0 µs
DEGR	None	-	-	-	-	-	-	81.5 µs	1.0 µs

Table Instructions

Table Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
FILL	V: Data Reg.	-	-	-	-	-	-	29.4 µs + 8.0 µs x N	1.0 µs
	V: Bit Reg.	-	-	-	-	-	-		-
	K: Constant	-	-	-	-	-	-	26.2 µs+ 8.0 µs x N	1.0 µs
	P: Indir. (Data)	-	-	-	-	-	-	55.1 µs + 8.0 µs x N	1.0 µs
	P: Indir. (Bit)	-	-	-	-	-	-		-
FIND	V: Data Reg. (N bits)	-	-	-	-	-	-	66.8 µs	1.0 µs
	V: Bit Reg. (N bits)	-	-	-	-	-	-	66.8 µs	1.0 µs
	K: Constant (N bits)	-	-	-	-	-	-	64.0 µs	1.0 µs
FDGT	V: Data Reg. (N bits)	-	-	-	-	-	-	66.1 µs	1.0 µs
	V: Bit Reg. (N bits)	-	-	-	-	-	-	66.1 µs	1.0 µs
	K: Constant (N bits)	-	-	-	-	-	-	55.2 µs	1.0 µs
FINDB	V: Data Reg. (N bits)	-	-	-	-	-	-	210.8 µs	1.0 µs
	V: Bit Reg. (N bits)	-	-	-	-	-	-	210.8 µs	1.0 µs
	P: Indir. (Data)	-	-	-	-	-	-	237.0 µs	1.0 µs
	P: Indir. (Bit)	-	-	-	-	-	-	237.0 µs	1.0 µs

Table Instructions (cont'd)

Table Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
MOV	Move V: data reg. to V: data reg	450 µs + 17 µs x N	6.2 µs	586 µs + 8 µs x N	8.4 µs	60.2 µs + 9.5 µs x N	0.9 µs	60.2 µs + 9.5 µs x N	0.9 µs
	Move V: bit reg. to V: data reg.	430 µs + 244 µs x N	6.2 µs	629 µs + 114.7 µs x N	8.4 µs				
	Move V: data reg to V: bit reg.	460 µs + 215 µs x N	6.2 µs	569 µs + 94.4 µs x N	8.4 µs				
	Move V: bit reg. to V:bit reg. N = #of words	490 µs + 448 µs x N	6.2 µs	693 µs + 198 µs x N	8.4 µs				
TTD	V: Data Reg.	-	-	-	-	-	-	66.9 µs	1.0 µs
	V: Bit Reg.							66.9 µs	1.0 µs
RFB	V: Data Reg.	-	-	-	-	-	-	66.8 µs	1.0 µs
	V: Bit Reg.							66.8 µs	1.0 µs
STT	V: Data Reg.	-	-	-	-	-	-	67.8 µs	1.0 µs
	V: Bit Reg.							67.8 µs	1.0 µs
	K: Constant							65.0 µs	1.0 µs
RFT	V: Data Reg.	-	-	-	-	-	-	51.1 µs	1.0 µs
	V: Bit Reg.							51.1 µs	1.0 µs
ATT	V: Data Reg.	-	-	-	-	-	-	53.5 µs	1.0 µs
	V: Bit Reg.							53.5 µs	1.0 µs
	K: Constant							50.8 µs	1.0 µs
TSHFL	V: Data Reg.	-	-	-	-	-	-	134.0 µs	1.0 µs
	V: Bit Reg.							134.0 µs	1.0 µs
TSHFR	V: Data Reg.	-	-	-	-	-	-	133.9 µs	1.0 µs
	V: Bit Reg.							133.9 µs	1.0 µs
ANDMOV	V: Data Reg.	-	-	-	-	-	-	80.2 µs	1.0 µs
	V: Bit Reg.							80.2 µs	1.0 µs
ORMOV	V: Data Reg.	-	-	-	-	-	-	80.4 µs	1.0 µs
	V: Bit Reg.							80.4 µs	1.0 µs
XORMOV	V: Data Reg.	-	-	-	-	-	-	80.4 µs	1.0 µs
	V: Bit Reg.							80.4 µs	1.0 µs
SWAP	V: Data Reg.	-	-	-	-	-	-	84.1 µs	1.0 µs
	V: Bit Reg.							84.1 µs	1.0 µs
SETBIT	V: Data Reg. (N bits)	-	-	-	-	-	-	59.5 µs	1.0 µs
	V: Bit Reg. (N bits)							59.5 µs	1.0 µs
RSTBIT	V: Data Reg. (N bits)	-	-	-	-	-	-	59.5 µs	1.0 µs
	V: Bit Reg. (N bits)							59.5 µs	1.0 µs

Table Instructions (cont'd)

Table Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
MOVMC	Move V: Data Reg. to EEPROM	-	-	356 µs + 7689 µs x N	8.4 µs	-	-	33.5 µs + 10.4 µs x N	0.9 µs
	Move V: Bit Reg. to EEPROM	-	-	392 µs + 7843 µs x N	8.4 µs	-	-	33.5 µs + 10.4 µs x N	0.9 µs
	Move from EEPROM to V: Data Reg.	250 µs + 201 µs x N	6.2 µs	520 µs + 181 µs x N	8.4 µs	50 µs + 15 µs x N	1.2 µs	33.5 µs + 10.4 µs x N	0.9 µs
	Move from EEPROM to V: Bit Reg. N=#of words	-	-	565 µs + 344 µs x N	8.4 µs	50 µs + 15 µs x N	1.2 µs	33.5 µs + 10.4 µs x N	0.9 µs
LDLBL	K	58 µs	8.4 µs	56 µs	8.4 µs	7.4 µs	1.5 µs	6.4 µs	1.3 µs

CPU Control Instructions

CPU Control Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
NOP	None	0 µs	0 µs	0 µs	0 µs	0.5 µs	0.5 µs	0.5 µs	0.5 µs
END	None	27 µs	27 µs	16 µs	16 µs	12.8 µs	0 µs	12.8 µs	0 µs
STOP	None	16 µs	5 µs	15 µs	7.4 µs	0 µs	0.9 µs	0 µs	0.9 µs
RSTWT	None	-	-	19 µs	8.4 µs	4.7 µs	0.9 µs	4.7 µs	0.9 µs

Program Control Instructions

Program Control Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
GOTO	K	-	-	14 µs	8.4 µs	5.1 µs	4.8 µs	5.1 µs	4.8 µs
LBL	K	-	-	0.6 µs	0.6 µs	5.7 µs	0.0 µs	5.7 µs	0.0 µs
FOR	V, K	-	-	32 µs	16.4 µs	85.8 µs	5.8 µs	85.8 µs	5.8 µs
NEXT	None	-	-	19 µs	0 µs	10.2 µs	0.0 µs	10.2 µs	0.0 µs
GTS	K	-	-	37 µs	11.4 µs	10.9 µs	5.5 µs	10.9 µs	5.5 µs
SBR	K	-	-	0.6 µs	0 µs	0.5 µs	0.0 µs	0.5 µs	0.0 µs
RT	None	-	-	35 µs	0 µs	9.9 µs	0.0 µs	9.9 µs	0.0 µs
RTC	None	-	-	-	-	-	-	11.4 µs	5.9 µs
MLS	K (1-7)	12 µs	12 µs	11.5 µs	11.5 µs	3.7 µs	3.7 µs	3.7 µs	3.7 µs
MLR	K (0-7) N=1 to 7	13 µs + 2.4 µs x N	13 µs + 2.4 µs x N	12.7 µs + 2.3 µs x N	12.7 µs + 2.3 µs x n	3.5 µs	3.5 µs	3.5 µs	3.5 µs

Interrupt Instructions

Interrupt Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
ENI	None	9 µs	5 µs	10.5 µs	8.4 µs	5.0 µs	1.0 µs	5.0 µs	1.0 µs
DISI	None	8 µs	5 µs	11 µs	8.4 µs	5.7 µs	0.9 µs	5.7 µs	0.9 µs
INT	O	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs
IRT	None	1.6 µs	0 µs	8 µs	0 µs	1.3 µs	0 µs	1.3 µs	0 µs
IRTC	None	-	-	-	-	-	-	0.5 µs	0 µs

Network Instructions

Network Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
RX	X, Y, C, T, CT, SP, S	-	-	TBD	TBD	251.3 µs	1.1 µs	251.3 µs	1.1 µs
	V: Data Reg.					251.3 µs	1.1 µs	251.3 µs	1.1 µs
	V: Bit Reg.					251.3 µs	1.1 µs	251.3 µs	1.1 µs
	P: Indir. (Data)					270.3 µs	1.9 µs	270.3 µs	1.9 µs
	P: Indir. (Bit)					270.3 µs	1.9 µs	270.3 µs	1.9 µs
WX	X, Y, C, T, CT, SP, S	-	-	TBD	TBD	252.0 µs	2.7 µs	252.0 µs	2.7 µs
	V: Data Reg.					252.0 µs	2.7 µs	252.0 µs	2.7 µs
	V: Bit Reg.					252.0 µs	2.7 µs	252.0 µs	2.7 µs
	P: Indir. (Data)					271.3 µs	3.4 µs	271.3 µs	3.4 µs
	P: Indir. (Bit)					271.3 µs	3.4 µs	271.3 µs	3.4 µs

Intelligent I/O Instructions

Intelligent I/O Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
RD	V: Data Reg.	TBD	TBD	TBD	TBD	385.7 µs	1.2 µs	385.7 µs	1.2 µs
	V: Bit Reg.					385.7 µs	1.2 µs	385.7 µs	1.2 µs
WT	V: Data Reg.	TBD	TBD	TBD	TBD	385.6 µs	1.2 µs	385.6 µs	1.2 µs
	V: Bit Reg.					385.6 µs	1.2 µs	385.6 µs	1.2 µs

Message Instructions

Message Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
FAULT	V: Data Reg.	171 µs	8.4 µs	23176 µs	8.4 µs	84.9 µs	1.1 µs	84.9 µs	1.1 µs
	V: Bit Reg.	253 µs	8.4 µs	23206 µs	8.4 µs	84.9 µs	1.1 µs	84.9 µs	1.1 µs
	K: Constant	2798 µs	8.4 µs	29108 µs	8.4 µs	80.8 µs	1.2 µs	80.8 µs	1.2 µs
DLBL	K	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs
NCON	K	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs
ACON	K	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs
PRINT	Text Data	-	-	-	-	36.3 µs	1.1 µs	36.3 µs	1.1 µs

RLL*PLUS* Instructions

RLL*PLUS* Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
ISG	S	31 µs	32 µs	28 µs	27 µs	20.9 µs	9.2 µs	20.9 µs	9.2 µs
SG	S	31 µs	32 µs	28 µs	27 µs	20.9 µs	9.2 µs	20.9 µs	9.2 µs
JMP	S	14 µs	8 µs	14.3 µs	8.4 µs	20.9 µs	3.7 µs	20.9 µs	3.7 µs
NJMP	S	14 µs	8 µs	13.3 µs	8.4 µs	21.0 µs	4.0 µs	21.0 µs	4.0 µs
CV	S	43 µs	27 µs	20 µs	20 µs	12.1 µs	12.1 µs	12.1 µs	12.1 µs
CVJMP	S (N stages, 1 to 16)	33 µs + 14.5 µs x N	23 µs	22.9 µs + 6.1 µs x N	10 µs	11.0 µs	11.0 µs	11.0 µs	11.0 µs
BCALL	C	18 µs	17 µs	17 µs	18 µs	22.1 µs	22.6 µs	22.1 µs	22.6 µs
BLK	C	32 µs	30 µs	17 µs	13 µs	17.1 µs	14.6 µs	17.1 µs	14.6 µs
BEND	None	17 µs	17 µs	9 µs	9 µs	8.7 µs	0.0 µs	8.7 µs	0.0 µs

DRUM Instructions

Drum Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
DRUM	CT	-	-	-	-	265.2 µs	48.8 µs	265.2 µs	48.8 µs
EDRUM	CT	-	-	-	-	189.5 µs	78.0 µs	189.5 µs	78.0 µs
MDRMD	CT	-	-	-	-	411.3 µs	216.4 µs	411.3 µs	216.4 µs
MDRMW	CT	-	-	-	-	378.6 µs	147.0 µs	378.6 µs	147.0 µs

Clock / Calender Instructions

Clock / Calender Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
DATE	V: Data Reg.	-	-	-	-	24.0 µs	1.2 µs	24.0 µs	1.2 µs
	V: Bit Reg.								
TIME	V: Data Reg.	-	-	-	-	50.8 µs	1.2 µs	50.8 µs	1.2 µs
	V: Bit Reg.								

Modbus Instructions

Modbus Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction		Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
MRX	Input, Input Register Coil, Holding Register	-	-	-	-	-	-	120.2 µs	1.3 µs
MWX	Input, Input Register Coil, Holding Register	-	-	-	-	-	-	21.3 µs	1.3 µs

ASCII Instructions

ASCII Instructions		D2-230		D2-240		D2-250-1		D2-260/D2-262	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
AIN	V	-	-	-	-	-	-	13.9 µs	12.0 µs
AFIND	V	-	-	-	-	-	-	111.5 µs	1.3 µs
AEX	V	-	-	-	-	-	-	111.7 µs	1.3 µs
CMPV	V	-	-	-	-	-	-	12.2 µs	1.3 µs
SWAPB	V	-	-	-	-	-	-	109.8 µs	1.3 µs
VPRINT	Text Data	-	-	-	-	-	-	161.6 µs	1.3 µs
PRINTV	V	-	-	-	-	-	-	163.3 µs	1.3 µs
ACRB	V	-	-	-	-	-	-	3.9 µs	1.1 µs

Special Relays

In This Appendix...

D2-230 CPU Special Relays

Startup and Real-Time Relays

SP0	First scan	ON for the first scan after a power cycle or program to run transition only. The relay is reset to off on the second scan. It is useful where a function needs to be performed only on program startup.
SP1	Always ON	Provides a contact to insure an instruction is executed every scan.
SP2	Always OFF	Provides a contact that is always off.
SP3	1 minute clock	ON for 30 seconds and off for 30 seconds.
SP4	1 second clock	ON for 0.5 second and off for 0.5 second.
SP5	100 ms clock	ON for 50ms and off for 50ms
SP6	50 ms clock	ON for 25ms and off for 25ms
SP7	Alternate scan	ON every other scan.

CPU Status Relays

SP12	Terminal run mode	ON when the CPU is in the run mode.
SP16	Terminal program mode	ON when the CPU is in the program mode.
SP20	Forced stop mode	ON when the STOP instruction is executed.
SP22	Interrupt enabled	ON when interrupts have been enabled using the ENI instruction.

System Monitoring

SP40	Critical error	ON when a critical error such as I/O communication loss has occurred.
SP41	Warning	ON when a non-critical error such as a low battery has occurred.
SP43	Battery low	ON when the CPU battery voltage is low (only if bit 12 of V7633 is set).
SP44	Program memory error	ON when a memory error such as a memory parity error has occurred.
SP45	I/O error	ON when an I/O error occurs. For example, an I/O module is withdrawn from the base, or an I/O bus error is detected.
SP47	I/O configuration error	ON if an I/O configuration error has occurred. The CPU power-up I/O configuration check must be enabled before this relay will be functional.
SP50	Fault instruction	ON when a Fault Instruction is executed.
SP51	Watch Dog timeout	ON if the CPU Watch Dog timer times out.
SP52	Grammatical error	ON if a grammatical error has occurred, either while the CPU is running or if the syntax check is run. V7755 will hold the exact error code.
SP53	Solve logic error	ON if CPU cannot solve the logic.

Accumulator Status

SP60	Value less than	ON when the accumulator value is less than the instruction value.
SP61	Value equal to	ON when the accumulator value is equal to the instruction value.
SP62	Greater than	ON when the accumulator value is greater than the instruction value.
SP63	Zero	ON when the result of the instruction is zero (in the accumulator).
SP64	Half borrow	ON when the 16-bit subtraction instruction results in a borrow.
SP65	Borrow	ON when the 32-bit subtraction instruction results in a borrow.
SP66	Half carry	ON when the 16-bit addition instruction results in a carry.
SP67	Carry	ON when the 32-bit addition instruction results in a carry.
SP70	Sign	ON anytime the value in the accumulator is negative.
SP71	Invalid octal number	ON when an Invalid octal number was entered. This also occurs when the V-memory specified by a pointer (P) is not valid.
SP73	Overflow	ON if overflow occurs in the accumulator when a signed addition or subtraction results in an incorrect sign bit.
SP74	Underflow	ON anytime a math operation results in an underflow error.
SP75	Data error	ON if a BCD number is expected and a non–BCD number is encountered.
SP76	Load zero	ON when any instruction loads a value of zero into the accumulator.

Counter Interface Module Relays

SP100	XO is ON	XO - ON when corresponding input is on.

Equal Relays for Multi-step Presets with Up/Down Counter #1 (for D2-230) (for use with the Counter Interface Module, D2-CTRINT)

SP540	Current = target value	ON when the counter current value equals the value in:	V2320
SP541			V2322
SP542			V2324
SP543			V2326
SP544			V2328
SP545			V2330
SP546			V2332
SP547			V2334
SP550			V2336
SP551			V2338
SP552			V2340
SP553			V2342
SP554			V2344
SP555			V2346
SP556			V2348
SP557			V2350
SP560			V2352
SP561			V2354
SP562			V2356
SP563			V2358
SP564			V2360
SP565			V2362
SP566			V2364
SP567			V2366
SP570			V2368
SP571			V2370

D2-240, D2-250-1, D2-260 and D2-262 CPU Special Relays

Startup and Real-Time Relays

SP0	First scan	ON for the first scan after a power cycle or program to run transition only. The relay is reset to off on the second scan. It is useful where a function needs to be performed only on program startup.
SP1	Always ON	Provides a contact to insure an instruction is executed every scan.
SP2	Always OFF	Provides a contact that is always off.
SP3	1 minute clock	ON for 30 seconds and off for 30 seconds.
SP4	1 second clock	ON for 0.5 second and off for 0.5 second.
SP5	100 ms clock	ON for 50ms and off for 50ms
SP6	50 ms clock	ON for 25ms and off for 25ms
SP7	Alternate scan	ON every other scan.

CPU Status Relays

SP11	Forced run mode	ON anytime the CPU switch is in the RUN position.
SP12	Terminal run mode	ON when the CPU switch is in the TERM position and the CPU is in the RUN mode.
SP13	Test run mode	ON when the CPU switch is in the TERM position and the CPU is in the test RUN mode.
SP14	Break Relay 1 (D2-250-1, D2-260 and D2-262)	ON when the BREAK instruction is executed. It is OFF when the CPU is in any other mode.
SP15	Test program mode	ON when the CPU is in the TERM position and the CPU is in the TEST PROGRAM MODE.
SP16	Terminal program mode	ON when the CPU switch is in the TERM position and the CPU is in the PROGRAM MODE.
SP17	Forced stop mode relay (D2-250-1, D2-260 and D2-262)	ON anytime the CPU keyswitch is in the STOP position.
SP20	Forced stop mode	ON when the STOP instruction is executed.
SP21	Break Relay 2 (D2-250-1, D2-260 and D2-262)	ON when the BREAK instruction is executed. It is OFF when the CPU mode is changed to RUN.
SP22	Interrupt enabled	ON when interrupts have been enabled using the ENI instruction.
SP25	CPU battery disabled relay (D2-250-1, D2-260 and D2-262)	ON when the CPU battery is disabled by special V-memory.

System Monitoring Relays

SP40	Critical error	ON when a critical error such as I/O communication loss has occurred.
SP41	Warning	ON when a non-critical error such as a low battery has occurred.
SP43	Battery low/dead	ON when the CPU battery voltage is low or dead. Note: The CPU must have a battery installed.
SP44	Program memory error	ON when a memory error such as a memory parity error has occurred.
SP45	I/O error	ON when an I/O error occurs. For example, an I/O module is withdrawn from the base, or an I/O bus error is detected.
SP46	Communications error	ON when a communications error has occurred on any of the CPU ports.
SP47	I/O configuration error	ON if an I/O configuration error has occurred. The CPU power-up I/O configuration check must be enabled before this relay will be functional.
SP50	Fault instruction	ON when a Fault Instruction is executed.
SP51	Watch Dog timeout	ON if the CPU Watch Dog timer times out.
SP52	Grammatical error	ON if a grammatical error has occurred either while the CPU is running or if the syntax check is run. V7755 will hold the exact error code.
SP53	Solve logic error	ON if CPU cannot solve the logic.
SP54	Intelligent I/O error	ON when communications with an intelligent module has occurred.
SP56	Table instruction overrun	ON if a table instruction with a pointer is executed and the pointer value is outside the table boundary

Accumulator Status Relays

SP53	Math/Table pointer error	ON if there is math execution error or a table pointer error.
SP60	Value less than	ON when the accumulator value is less than the instruction value.
SP61	Value equal to	ON when the accumulator value is equal to the instruction value.
SP62	Greater than	ON when the accumulator value is greater than the instruction value.
SP63	Zero	ON when the result of the instruction is zero (in the accumulator).
SP64	Half borrow	ON when the 16-bit subtraction instruction results in a borrow.
SP65	Borrow	ON when the 32-bit subtraction instruction results in a borrow.
SP66	Half carry	ON when the 16-bit addition instruction results in a carry.
SP67	Carry	ON when the 32-bit addition instruction results in a carry.
SP70	Sign	ON anytime the value in the accumulator is negative.
SP71	Invalid octal number	ON when an Invalid octal number was entered. This also occurs when the V-memory specified by a pointer (P) is not valid.
SP72	Floating Point	ON when the numerical value in the accumulator is a floating point number.
SP73	Overflow	ON if overflow occurs in the accumulator when a signed addition or subtraction results in an incorrect sign bit.
SP74	Under flow	ON when a floating point math operation results in an underflow error.
SP75	Data error	ON if data is not a numerical value.
SP76	Load zero	ON when any instruction loads a value of zero into the accumulator.

Counter Interface Module Relays

SP100	X0 is ON	X0 - ON when corresponding input is ON.
SP101	X1 is ON	X1 - ON when corresponding input is ON.
SP102	X2 is ON	X2 - ON when corresponding input is ON.
SP103	X3 is ON	X3 - ON when corresponding input is ON.

Communications Monitoring Relays

SP116	D2-240 CPU communication	ON when the CPU is communicating with another device
SP116	D2-250-1, D2-260 and D2-262 communication	ON when Port 2 is communicating with another device
SP117	Comm error Port 2 (D2-250-1/D2-260/D2-262)	ON when Port 2 has encountered a communication error.
SP120	Module busy Slot 0	ON when the communication module in slot 0 is busy transmitting or receiving. You must use this relay with the RX or WX instructions to prevent attempting to execute a RX or WX while the module is busy.
SP121	Comm error Slot 0	ON when the communication module in slot 0 of the local base has encountered a communication error.
SP122	Module busy Slot 1	ON when the communication module in slot 1 of the local base is busy transmitting or receiving. You must use this relay with the RX or WX instructions to prevent attempting to execute a RX or WX while the module is busy.
SP123	Comm error Slot 1	ON when the communication module in slot 1 of the local base has encountered a communication error.
SP124	Module busy Slot 2	ON when the communication module in slot 2 of the local base is busy transmitting or receiving. You must use this relay with the RX or WX instructions to prevent attempting to execute a RX or WX while the module is busy.
SP125	Comm error Slot 2	ON when the communication module in slot 2 of the local base has encountered a communication error.
SP126	Module busy Slot 3	ON when the communication module in slot 3 of the local base is busy transmitting or receiving. You must use this relay with the RX or WX instructions to prevent attempting to execute a RX or WX while the module is busy.
SP127	Comm error Slot 3	ON when the communication module in slot 3 of the local base has encountered a communication error.
SP130	Module busy Slot 4	ON when the communication module in slot 4 of the local base is busy transmitting or receiving. You must use this relay with the RX or WX instructions to prevent attempting to execute a RX or WX while the module is busy.
SP131	Comm error Slot 4	ON when the communication module in slot 4 of the local base has encountered a communication error.
SP132	Module busy Slot 5	ON when the communication module in slot 5 of the local base is busy transmitting or receiving. You must use this relay with the RX or WX instructions to prevent attempting to execute a RX or WX while the module is busy.
SP133	Comm error Slot 5	ON when the communication module in slot 5 of the local base has encountered a communication error.
SP134	Module busy Slot 6	ON when the communication module in slot 6 of the local base is busy transmitting or receiving. You must use this relay with the RX or WX instructions to prevent attempting to execute a RX or WX while the module is busy.
SP135	Comm error Slot 6	ON when the communication module in slot 6 of the local base has encountered a communication error.
SP136	Module busy Slot 7	ON when the communication module in slot 7 of the local base is busy transmitting or receiving. You must use this relay with the RX or WX instructions to prevent attempting to execute a RX or WX while the module is busy.
SP137	Comm error Slot 7	ON when the communication module in slot 7 of the local base has encountered a communication error.

Equal Relays for Multi-step Presets with Up/Down Counter #1 (supported by D2-240, D2-250-1 and D2-260).* For use with the Counter Interface Module D2-CTRINT

SP540	Current = target value	ON when the counter current value equals the value in:	V3630
SP541			V3632
SP542			V3634
SP543			V3636
SP544			V3640
SP545			V3642
SP546			V3644
SP547			V3646
SP550			V3650
SP551			V3652
SP552			V3654
SP553			V3656
SP554			V3660
SP555			V3662
SP556			V3664
SP557			V3666
SP560			V3670
SP561			V3672
SP562			V3674
SP563			V3676
SP564			V3700
SP565			V3702
SP566			V3704
SP567			V3706

* Not supported by D2-262

Equal Relays for Multi-step Presets with Up/Down Counter #2 (supported by D2-240, D2-250-1 and D2-260).* For use with the Counter Interface Module D2-CTRINT

SP570	Current = target value	ON when the counter current value equals the value in:	V3710
SP571			V3712
SP572			V3714
SP573			V3716
SP574			V3720
SP575			V3722
SP576			V3724
SP577			V3726
SP600			V3730
SP601			V3732
SP602			V3734
SP603			V3736
SP604			V3740
SP605			V3742
SP606			V3744
SP607			V3746
SP610			V3750
SP611			V3752
SP612			V3754
SP613			V3756
SP614			V3760
SP615			V3762
SP616			V3764
SP617			V3766

* Not supported by D2-262

PLC Memory

In This Appendix...

DL205 PLC Memory

When designing a PLC application, it is important for the PLC user to understand the different types of memory in the PLC. The DL205 CPUs use two types of memory: RAM and EEPROM. RAM is Random Access Memory and EEPROM is Electrically Erasable Programmable Read Only Memory. The PLC program is stored in EEPROM, and the PLC V-memory data is stored in RAM. There is also a small range of V-memory that can be copied to EEPROM, which will be explained later.

The V-memory in RAM can be configured as either retentive or non-retentive.

Retentive memory is memory that is configured by the user to maintain values through a power cycle or a PROGRAM to RUN transition. Non-retentive memory is memory that is configured by the PLC user to clear data after a power cycle or a PROGRAM to RUN transition. The retentive ranges can be configured with either the Handheld Programmer using AUX57 or *Direct*SOFT (PLC > Setup).

The contents of RAM memory can be written to and read from an infinite number of times, but RAM requires a power source to maintain the contents of memory. The contents of RAM are maintained by the internal power supply (5VDC) only while the PLC is powered by an external source, normally 120VAC. When power to the PLC is turned off, the contents of RAM are maintained by a "Super-Capacitor." If the Super-Capacitor ever discharges, the contents of RAM will be lost. In a D2-230 and D2-240 the data retention time of the Super-Capacitor backed RAM is 3 weeks maximum, and 4 1/2 days minimum (at 60°C). An optional battery, (ADC p/n D2-BAT), can be added to maintain RAM retentive memory if the D2-230 or D2-240 is ever without external power (see Volume I, page 3-14 for a detailed explanation).

In a D2-250-1 and D2-260, the Super-Capacitor backed RAM will be retained for 15.9 hours and in a D2-262, it is retained for 1.9 hours. The D2-260 uses a larger capacitor that provides current to the external real-time clock. The D2-262 has a smaller capacitor that provides current to the micro-processing unit with the integrated real-time clock, which results in the D2-262 hardware pulling more current and having less memory retention time. **Therefore, the purpose of the capacitor in the D2-262 is for exchanging the battery, not for memory retention**.

The contents of EEPROM memory can be read from an infinite number of times, but there is a limit to the number of times it can be written to (typical specification is 100,000 writes). EEPROM does not require a power source to maintain the memory contents. It will retain the contents of memory indefinitely.

PLC user V-memory is stored in both volatile RAM and non-volatile EEPROM memory. The table below shows the memory areas for each DL205 CPU.

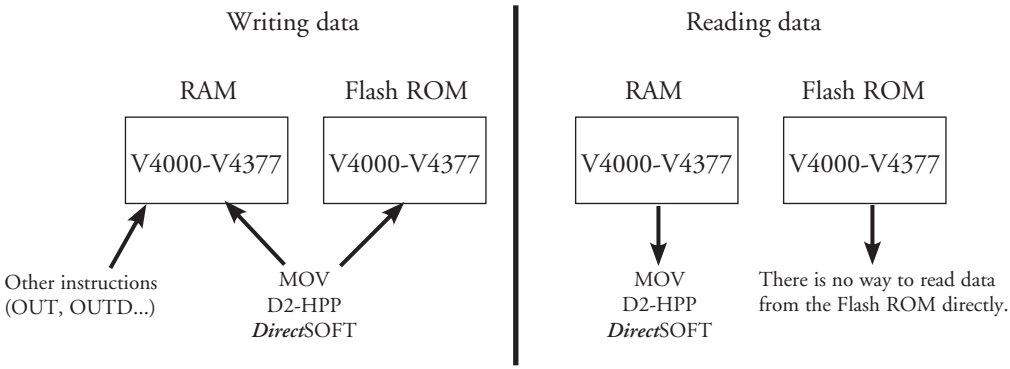
DL205 Memory Area				
PLC Type	**D2-230**	**D2-240**	**D2-250-1**	**D2-260/D2-262**
Volatile RAM	V2000–V2377	V2000–V3777	V1400–V7377 V10000–V17777	V400–V777 V1400–V7377 V10000–V35777
Non-volatile	V4000–V4177	V4000–V4377	-	-

Data values that must be retained for long periods of time, when the PLC is powered off, should be stored in EEPROM-based V-memory. Since EEPROM is limited to the number of times it can be written to, it is suggested that transitional logic, such as a one-shot, be used to write the data one time, instead on each CPU scan.

Data values that are continually changing or which can be initialized with program logic should be stored in RAM-based V-memory.

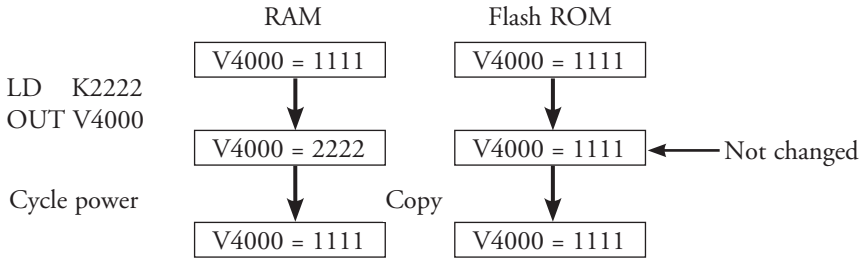
Non-volatile V-memory in the DL205

Two types of memory are assigned for the non-volatile V-memory area: RAM and flash ROM (EEPROM). They are sharing the same V-memory addresses; however, **you can only use the MOV instruction, D2-HPP and *Direct*SOFT to write data to the flash ROM.** When you write data to the flash ROM, the same data is also written to RAM. If you use other instructions, you can only write data to RAM. When you read data from the non-volatile V-memory area, the data is always read from RAM. The following explanation uses the D2-240 CPU as an example.



After a power cycle, the PLC always copies the data in the flash ROM to the RAM.

If you use the instructions except for the MOV instruction to write data into the non-volatile V-memory area, you only update the data in RAM. After a power cycle, the PLC copies the previous data from the flash memory to the RAM, so you may think the data you changed has disappeared. To avoid trouble such as this, we recommend that you use the MOV instruction.



This appears to be previous data returning.

PLC Memory Processes

The following tables show how memory backup, memory restoration and retentive memory are handled by the D2-250-1, D2-260 and D2-262.

Backup

Moving SRAM to Flash ROM

- NO - Backup does not take place
- YES - see table below

Backup - Moving SRAM to Flash ROM						
	Power ON	RUN to STOP	STOP	RUN	User Memory	ScratchPad
D2-250-1 / D2-260	Yes	Yes	Yes	No	V7400-V7577	Yes
D2-262	No	Yes	Yes	No	V7620 – V7627 V7720 – V7722 V7630 – V7641 V7650 – V7656 V7740 – V7742	Yes

Restore

Moving Flash ROM to SRAM. Has a backup ever occurred:

- NO - restore does not take place
- YES - see table below

Restore - Moving Flash ROM to SRAM				
	Power ON	RUN to STOP	User Memory	ScratchPad
D2-250-1 / D2-260	Yes	No	V7400-V7577	Yes
D2-262	Yes	No	V7620 – V7627 V7720 – V7722 V7630 – V7641 V7650 – V7656 V7740 – V7742	Yes

Retentive Memory

Maintains SRAM during Power Off condition (battery installed)

Retentive Memory			
	Memory (if selected)	**Capacitor Hold Time (No battery present)**	**Real Time Clock**
D2-250-1 / D2-260	V0-V37777 T0-T377 CT0-CT377 C0-C3777 S0-S1777	~15 hours	External from CPU
D2-262	V0-V37777 T0-T377 CT0-CT377 C0-C3777 S0-S1777	~1.5 hours	Built-in CPU

WARNING: Super-capacitor provides enough time to change the battery, it is not meant to retain memory. Install the optional battery, D2-BAT-1, if you have concerns about losing retentive data.

DL205 Product Weight Table

In This Appendix...

DL205 Product Weight Table

CPUs	Weight
D2-230	2.8 oz. (80g)
D2-240	2.8 oz. (80g)
D2-250-1	2.5 oz. (70g)
D2-260	2.5 oz. (70g)
D2-262	2.5 oz. (70g)
I/O Bases	
D2-03B-1	12.3 oz. (350g)
D2-03BDC1-1	11.4 oz. (322g)
D2-03BDC-2	10.1oz. (285g)
D2-04B-1	13.4 oz. (381g)
D2-04BDC1-1	12.5 oz. (354g)
D2-04BDC-2	11.2 oz. (317g)
D2-06B-1	14.4 oz. (410g)
D2-06BDC1-1	13.8 oz. (392g)
D2-06BDC2-1	13.8 oz. (392g)
D2-09B-1	18.6 oz. (530g)
D2-09BDC1-1	18.3 oz. (522g)
D2-09BDC2-1	19 oz. (530g)
DC Input Modules	
D2-08ND3	2.3 oz. (65g)
D2-16ND3-2	2.3 oz. (65g)
D2-32ND3	2.1oz. (60g)
D2-32ND3-2	2.1oz. (60g)
AC Input Modules Weight	
D2-08NA-1	2.5 oz. (70g)
D2-08NA-2	2.5 oz. (70g)
D2-16NA	2.4 oz. (68g)
DC Input/Relay Output Module	
D2-08CDR	3.5 oz. (100g)

DC Output Modules	
D2-04TD1	2.8 oz. (80g)
D2-08TD1	2.3 oz. (65g)
D2-08TD2	2.1 oz. (60g)
D2-16TD1-2	2.3 oz. (65g)
D2-16TD2-2	2.8 oz. (80g)
F2-16TD1P	2.0 oz. (56g)
F2-16TD2P	2.0 oz. (56g)
D2-32TD1	2.1 oz. (60g)
D2-32TD2	2.1 oz. (60g)
AC Output Modules	
D2-08TA	2.8 oz. (80g)
F2-08TA	3.5 oz. (99g)
D2-12TA	2.8 oz. (80g)
Relay Output Modules	
D2-04TRS	2.8 oz. (80g)
D2-08TR	3.9 oz. (114g)
D2-12TR	4.6 oz. (130g)
F2-08TR	5.5 oz. (156g)
F2-08TRS	5.5 oz. (156g)
CPU-Slot Controllers	
H2-EBC	1.6 oz. (45g)
H2-EBC-F	2.1 oz. (60g)
F2-SDS-1	2.8 oz. (80g)
H2-PBC	2.1 oz. (60g)
F2-DEVNETS-1	3.0 oz. (86g)

Analog Modules Weight	
F2-04AD-1	3.0 oz. (86g)
F2-04AD-2	3.0 oz. (86g)
F2-04AD-1L	3.0 oz. (86g)
F2-04AD-2L	3.0 oz. (86g)
F2-08AD-1	3.0 oz. (86g)
F2-08AD-2	4.2 oz. (118g)
F2-02DA-1	2.8 oz. (80g)
F2-02DA-2	2.8 oz. (80g)
F2-02DA-1L	2.8 oz. (80g)
F2-02DA-2L	2.8 oz. (80g)
F2-08DA-1	2.8 oz. (80g)
F2-08DA-2	3.8 oz. (109g)
F2-02DAS-1	3.8 oz. (109g)
F2-02DAS-2	3.8 oz. (109g)
F2-4AD2DA	4.2 oz. (118g)
F2-8AD4DA-1	4.2 oz. (118g)
F2-8AD4DA-2	4.2 oz. (118g)
F2-04RTD	3.0 oz (86g)
F2-04THM	3.0 oz (86g)
Specialty Modules	
H2-CTRIO	2.3 oz. (65g)
H2-CTRIO2	2.2 oz. (62g)
D2-CTRINT	2.3 oz. (65g)
H2-ECOM	1.6 oz. (45g)
H2-ECOM100	1.5 oz. (43g)
H2-ECOM-F	5.5 oz. (156g)
H2-ERM(100)	1.6 oz. (45g)
H2-ERM-F	5.5 oz. (156g)
H2-SERIO	1.5 oz. (43g)
D2-DCM	3.8 oz. (109g)
F2-CP128	3.9 oz. (111g)
D2-EM	2.3 oz. (65g)
D2-CM	1.8 oz. (50g)
F2-08SIM	2.5 oz. (70g)

ASCII Table

In This Appendix...

ASCII Conversion Table

DECIMAL TO HEX TO ASCII CONVERTER											
DEC	HEX	ASCII	DEC	HEX	ASCII	DEC	HEX	ASCII	DEC	HEX	ASCII
0	00	NUL	32	20	space	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	“	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	‘	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	TAB	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Numbering Systems

In This Appendix...

Introduction

As almost anyone who uses a computer is somewhat aware, the actual operations of a computer are done with a binary number system. Traditionally, the two possible states for a binary system are represented by the digits "zero" (0) and "one" (1), although "off" and "on" or sometimes "no" and "yes" are closer to what is actually involved. Most of the time a typical PC user has no need to think about this aspect of computers, but every now and then one gets confronted with the underlying nature of the binary system.

A PLC user, specifically the PLC programmer, should be more aware of the binary system. This appendix will provide an explanation of the numbering systems most commonly used by a PLC.

Binary Numbering System

Computers, including PLCs, use the Base 2 numbering system, which is called Binary and often called Decimal. Much the same, a PLC relies on only two valid digits, zero and one, or off and on respectively. You would think that it would be hard to have a numbering system built on Base 2 with only two possible values, but the secret is by encoding using several digits.

Each digit in the base 2 system, when referenced by a computer, is called a bit. When four bits are grouped together, they form what is known as a nibble. Eight bits or two nibbles would be a byte. Sixteen bits or two bytes would be a word (Table 1). Thirty-two bits or two words is a double word.

Word															
Byte								Byte							
Nibble				Nibble				Nibble				Nibble			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1

Binary is not "natural" for us to use since we grow up using the base 10 system. Base 10 uses the numbers 0-9. From now on, the different bases will be shown as a subscripted number following the number. Example; 10 decimal would be 10_{10}.

Table 2 shows how base 2 numbers relate to their decimal equivalents.

A nibble of 1001_2 would be equal to a decimal number 9 ($1*2^3 + 1*2^0$ or $8_{10} + 1_{10}$). A byte of 11010101_2 would be equal to 213 ($1*2^7 + 1*2^6 + 1*2^4 + 1*2^2 + 1*2^0$ or $128_{10} + 64_{10} + 16_{10} + 4_{10} + 1_{10}$).

Binary/Decimal Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Power	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal Bit Value																
Max Value	65535_{10}															

Table 2

Hexadecimal Numbering System

The binary numbering system can be difficult and cumbersome to interpret for some users. Therefore, the hexadecimal numbering system was developed as a convenience for humans since the PLC (computer) only understands pure binary. The hexadecimal system is useful because it can represent every byte (8 bits) as two consecutive hexadecimal digits. It is easier for us to read hexadecimal numbers than binary numbers.

The hexadecimal numbering system uses 16 characters (base 16) to represent values. The first ten characters are the same as our decimal system, 0-9, and the first six letters of the alphabet, A-F. Table 3 lists the first eighteen decimal numbers; 0-17 in the left column and the equivalent hexadecimal numbers are shown in the right column.

Decimal	Hex	Decimal	Hex
0	0	9	9
1	1	10	A
2	2	11	B
3	3	12	C
4	4	13	D
5	5	14	E
6	6	15	F
7	7	16	10
8	8	17	11

Table 3

Note that "10" and "11" in hex are not the same as "10" and "11" in decimal. Only the first ten numbers 0-9 are the same in the two representations. For example, consider the hex number "D8AF." To evaluate this hex number, use the same method used to write decimal numbers. Each digit in a decimal number represents a multiple of a power of ten (base 10). Powers of ten increase from right to left. For example, the decimal number 365 means $3x10^2 + 6x10 + 5$. In hex each digit represents a multiple of a power of sixteen (base 16). Therefore, the hex number D8AF translated to decimal means $13x16^3 + 8x16^2 + 10x16 + 15 = 55471$. However, going through the arithmetic for hex numbers in order to evaluate them is not really necessary. The easier way is to use the calculator that comes as an accessory in Windows. It can convert between decimal and hex when in "Scientific" view.

Note that a hex number such as "365" is not the same as the decimal number "365." Its actual value in decimal terms is $3x16^2 + 6x16 + 5 = 869$. To avoid confusion, hex numbers are often labeled or tagged so that their meaning is clear. One method of tagging hex numbers is to append a lower case "h" at the end. Another method of labeling is to precede the number with 0x. Thus, the hex number "D8AF" can also be written "D8AFh." where the lower case "h" at the end is just a label to make sure we know that it is a hex number. Also, D8AF can be written with a labeling prefix as "0xD8AF."

Octal Numbering System

Many of the early computers used the octal numbering system for compiled printouts. Today, the PLC is about the only device that uses the Octal numbering system. The octal numbering system uses eight values to represent numbers. The values are 0-7 being Base 8. Table 4 shows the first 31 decimal digits in octal. Note that the octal values are 0-7, 10-17, 20-27, and 30-37.

Octal	Decimal	Octal	Decimal
0	0	20	16
1	1	21	17
2	2	22	18
3	3	23	19
4	4	24	20
5	5	25	21
6	6	26	22
7	7	27	23
10	8	30	24
11	9	31	25
12	10	32	26
13	11	33	27
14	12	34	28
15	13	35	29
16	14	36	30
17	15	37	31

Table 4

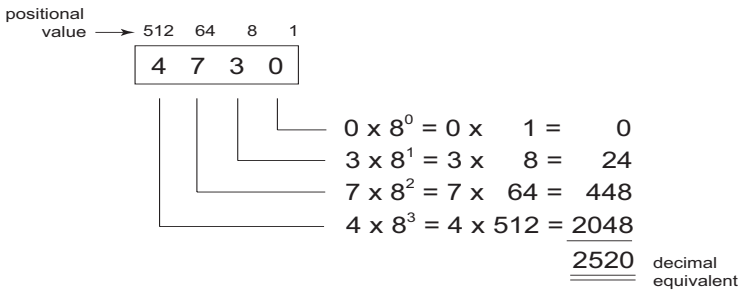
This follows the *Direct*LOGIC PLCs. Refer to Chapter 3 bit maps and notice that the memory addresses are numbered in octal, as well as each bit. The octal system is much like counting in the decimal system without the digits 8 and 9 being available.

The general format for four digits of the octal number is:

$$(d \times 8^0) + (d \times 8^1) + (d \times 8^2) + (d \times 8^3)$$

where "d" means digit. This is the same format used in the binary, decimal, or hexadecimal systems except that the base number for octal is 8.

Using the powers of expansion, the example below shows octal 4730 converted to decimal.



Binary Coded Decimal (BCD) Numbering System

BCD is a numbering system where four bits are used to represent each decimal digit. The binary codes corresponding to the hexadecimal digits A-F are not used in the BCD system. For this reason numbers cannot be coded as efficiently using the BCD system. For example, a byte can represent a maximum of 256 different numbers (i.e., 0-255) using normal binary, whereas only 100 distinct numbers (i.e., 0-99) could be coded using BCD. Also, note that BCD is a subset of hexadecimal and neither one does negative numbers.

BCD Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Power	10^3				10^2				10^1				10^0			
Bit Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Max Value	9				9				9				9			

Table 5

One plus for BCD is that it reads like a decimal number, whereas 867 in BCD would mean 867 decimal. No conversion is needed; however, within the PLC, BCD calculations can be performed if numbers are adjusted to BCD after normal binary arithmetic.

Real (Floating Point) Numbering System

The terms Real and floating-point both describe IEEE-754 floating point arithmetic. This standard specifies how single precision (32-bit) and double precision (64-bit) floating point numbers are to be represented as well as how arithmetic should be carried out on them. Most PLCs use the 32-bit format for floating point (or Real) numbers which will be discussed here.

	Real (Floating Point 32) Bit Pattern															
Bit #	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Sign	Exponent								Mantissa						
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Mantissa (continues from above)															

Table 6

Floating point numbers which *Direct*LOGIC PLCs use have three basic components: sign, exponent and mantissa. The 32-bit word required for the IEEE standard floating point numbers is shown in Table 6. It is represented as a number from 0 to 31, left to right. The first bit (31) is the sign bit, the next eight bits (30-23) are the exponent bits and the final 23 bits (22-0) are the fraction bits.

In summary:

The sign bit is either "0" for positive or "1" for negative;

The exponent uses base 2;

The first bit of the mantissa is typically assumed to be "1.*fff*", where "*f*" is the field of fraction bits.

The Internet can provide a more in-depth explanation of the floating point numbering system. One website to look at is:

http://www.psc.edu/general/software/packages/ieee/ieee.html

BCD/Binary/Decimal/Hex/Octal - What is the Difference?

Sometimes there is confusion about the differences between the data types used in a PLC. The PLC's native data format is BCD, while the I/O numbering system is octal. Other numbering formats used are binary and Real. Although data is stored in the same manner (0s and 1s), there are differences in the way that the PLC interprets it.

While all of the formats rely on the base 2 numbering system and bit-coded data, the format of the data is dissimilar. Table 7 below shows the bit patterns and values for various formats.

Binary/Decimal Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Bit Value	32678	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Max Value	65535															
Hexadecimal Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Bit Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Max Value	F				F				F				F			
BCD Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Bit Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Max Value	9				9				9				9			
Octal Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Value	1	4	2	1	4	2	1	4	2	1	4	2	1	4	2	1
Max Value	1	7			7			7			7			7		
Real (Floating Point 32) Pattern																
Bit #	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Sign	Exponent								Mantissa						
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Mantissa (continued from above)															

Table 7

As seen in Table 7, the BCD and hexadecimal formats are similar, although the maximum number for each grouping is different (9 for BCD and F for hexadecimal). This allows both formats to use the same display method. The unfortunate side effect is that unless the data type is documented, it's difficult to know what the data type is unless it contains the letters A-F.

Data Type Mismatch

Data type mismatching is a common problem when using an operator interface. Diagnosing it can be a challenge until you identify the symptoms. Since the PLC uses BCD as the native format, many people tend to think it is interchangeable with binary (unsigned integer) format. This is true to some extent, but not in this case. Table 8 shows how BCD and binary numbers differ.

Data Type Mismatch												
Decimal	0	1	2	3	4	5	6	7	8	9	10	11
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	0001 0000	0001 0001
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	0000 1010	0000 1011

Table 8

As the table shows, BCD and binary share the same bit pattern up until you get to the decimal number 10. Once you get past 10, the bit pattern changes. The BCD bit pattern for the decimal 10 is actually equal to a value of 16 in binary, causing the number to jump six digits when viewing it as BCD. With larger numbers, the error multiplies. Binary values from 10 to 15 Decimal are actually invalid for the BCD data type.

Looking at a larger number, such as the value shown in Table 9, both the BCD bit pattern and the decimal bit pattern correspond to a base 10 value of 4095_{10}. If bit patterns are read, or interpreted, in a different format than what is used to write them, the data will not be correct. For instance, if the BCD bit pattern is interpreted as a decimal (binary) bit pattern, the result is a base 10 value of 16533_{10}. Similarly, if you try to view the decimal (binary) bit pattern as a BCD value, it is not a valid BCD value at all, but could be represented in hexadecimal as 0xFFF.

Look at the following example and note the same value represented by the different numbering systems.

Base 10 Value	BCD Bit Pattern	Binary Bit Pattern
4095	0100 0000 1001 0101	1111 1111 1111

Table 9

0100	0011	Binary		0001	0010	0011	0100	Binary
6	7	Decimal		4	6	6	0	Decimal
4	3	Hex		1	2	3	4	Hex
0110	0111	BCD		0100	0110	0110	0000	BCD
1	0 3	Octal		1	1	0	6	4 Octal

Signed vs Unsigned Integers

So far, we have dealt with unsigned data types only. Now we will deal with signed data types (negative numbers). The BCD and hexadecimal numbering systems do not use signed data types.

In order to signify that a number is negative or positive, we must assign a bit to it. Usually, this is the Most Significant Bit (MSB) as shown in Table 10. For a 16-bit number, this is bit 15. This means that for 16-bit numbers we have a range of -32767 to 32767.

Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSB															LSB

Table 10

There are two ways to encode a negative number: two's complement and Magnitude Plus sign. The two methods are not compatible.

The simplest method to represent a negative number is to use one bit of the PLC word as the sign of a number while the remainder of the word gives its magnitude. It is general convention to use the most significant bit (MSB) as the sign bit: a 1 will indicate a negative, and a 0 will indicate a positive number. Thus, a 16-bit word allows numbers in the range ±32767. Table 11 shows a representation of 100 and a representation of -100 in this format.

Magnitude Plus Sign	
Decimal	**Binary**
100	0000 0000 0110 0100
-100	1000 0000 0110 0100

Table 11

Two's complement is a bit more complicated. A simple formula for two's complement is to invert the binary and add one (see Table 12). Basically, 1s are being changed to 0s and all 0s are being changed to 1.

Two's Complement	
Decimal	**Binary**
100	0000 0000 0110 0100
-100	1111 1111 1001 1011

Table 12

More information about 2's complement can be found on the Internet at the following website:
https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html#fromtwo

AutomationDirect.com Products and Data Types

DirectLOGIC PLCs

The *Direct*LOGIC PLC family uses the octal numbering system for all addressing which includes: inputs, outputs, internal V-memory locations, timers, counters, internal control relays (bits), etc. Most data in the PLC, including timer and counter current values, is in BCD format by default. User data in V-memory locations may be stored in other data types if it is changed by the programmer, or comes from some external source, such as an operator interface. Any manipulation of data must use instructions appropriate for that data type which includes: Load instructions, Math instructions, Out box instructions, comparison instructions, etc. In many cases, the data can be changed from one data type to another, but be aware of the limitations of the various data types when doing so. For example, to change a value from BCD to decimal (binary), use a BIN instruction box. To change from BCD to a real number, use a BIN and a BTOR instruction box. When using Math instructions, ***the data types must match.*** For example, a BCD or decimal (binary) number cannot be added to a real number, and a BCD number cannot be added to a decimal (binary) number. If the data types are mismatched, the results of any math operation will be meaningless.

To simplify number conversions, Intelligent Box (IBox) Instructions are available with *Direct*SOFT. These instruction descriptions are located in Volume 1, page 5-230, in the Math IBox group.

Most *Direct*LOGIC analog modules can be set up to give the raw data in decimal (binary) format or in BCD format, so it is necessary to know how the module is being used. *Direct*LOGIC PID is another area where not all values are in BCD. In fact, nearly all of the PID parameters are stored in the PLC memory as decimal (binary) numbers.

NOTE: *The PID algorithm uses magnitude plus sign for negative decimal (binary) numbers, whereas the standard math functions use two's complement. This can cause confusion while debugging a PID loop.*

When using the Data View in *Direct*SOFT, be certain that the proper format is selected for the element to be viewed. The data type and length are selected using the drop-down boxes at the top of the Data View window. Also notice that BCD is called BCD/Hex. Remember that BCD is a subset of hexadecimal so they share a display format even though the values may be different. This is where good documentation of the data type stored in memory is crucial.

C-more/C-more Micro-Graphic Panels

In the C-more and C-more Micro-Graphic HMI operator panels, the 16-bit BCD format is listed as "BCD int 16." Binary format is either "Unsigned int 16" or "Signed int 16" depending on whether or not the value can be negative. Real number format is "Floating PT 32."

More available formats are, "BCD int 32", "Unsigned int 32" and "Signed int 32."

European Union Directives (CE)

In This Appendix...

European Union (EU) Directives

NOTE: *The information contained in this section is intended as a guideline and is based on our interpretation of the various standards and requirements. Since the actual standards are issued by other parties, and in some cases governmental agencies, the requirements can change over time without advance warning or notice. Changes or additions to the standards can possibly invalidate any part of the information provided in this section.*

This area of certification and approval is absolutely vital to anyone who wants to do business in Europe. One of the key tasks that faced the EU member countries and the European Economic Area (EEA) was the requirement to bring several similar yet distinct standards together into one common standard for all members. The primary purpose of a single standard was to make it easier to sell and transport goods between the various countries and to maintain a safe working and living environment. The Directives that resulted from this merging of standards are now legal requirements for doing business in Europe. Products that meet these Directives are required to have a CE mark to signify compliance.

Member Countries

As of January 1, 2015, the members of the EU are Austria, Belgium, Bulgaria, Croatia, Republic of Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, and United Kingdom. Iceland, Liechtenstein, and Norway together with the EU members make up the European Economic Area (EEA) and all are covered by the Directives.

Applicable Directives

There are several Directives that apply to our products. Directives may be amended, or added, as required.

- **Electromagnetic Compatibility (EMC) Directive** — this Directive attempts to ensure that devices, equipment, and systems have the ability to function satisfactorily in an electromagnetic environment without introducing intolerable electromagnetic disturbance to anything in that environment.
- **Machinery Safety Directive** — this Directive covers the safety aspects of the equipment, installation, etc. There are several areas involved, including testing standards covering both electrical noise immunity and noise generation.
- **Low Voltage Directive (LVD)** — this Directive is also safety related and covers electrical equipment that has voltage ranges of 50–1000VAC and/or 75–1500VDC.
- **Battery Directive** — this Directive covers the production, recycling, and disposal of batteries.

Compliance

NOTE: *As of July 22, 2017 ROHS has been added as an additional requirement for CE Compliance per Directive 2011/65/EU. All products bearing the CE mark must be ROHS compliant.*

Certain standards within each Directive already require mandatory compliance. The EMC Directive, which has gained the most attention, became mandatory as of January 1, 1996. The Low Voltage Directive became mandatory as of January 1, 1997.

Ultimately, we are all responsible for our various pieces of the puzzle. As manufacturers, we must test our products and document any test results and/or installation procedures that are necessary to comply with the Directives. As an end user, you are responsible for installing these products in a manner which will ensure compliance with the applicable standards is maintained. You are also responsible for testing any combinations of products that may (or may not) comply with the Directives when used together. The end user of the products must comply with any Directives that may cover maintenance, disposal, etc. of equipment or various components. *Although we strive to provide the best assistance available, it is impossible for us to test all possible configurations of our products with respect to any specific Directive. Because of this, it is ultimately your responsibility to ensure that your machinery (as a whole) complies with these Directives and to keep up with applicable Directives and/or practices that are required for compliance.*

As of January 1, 1999, the DL05, DL06, DL205, DL305, and DL405 PLC systems (**except for the D2-262**) manufactured by Koyo Electronics Industries, FACTS Engineering or HOST Engineering, when properly installed and used, conform to the Electromagnetic Compatibility (EMC), Low Voltage Directive, and Machinery Directive requirements of the following standards.

- **EMC Directive Standards Relevant to PLCs:**

  EN50081–1 Generic emission standard for residential, commercial, and light industry

  EN50081–2 Generic emission standard for industrial environment

  EN50082–1 Generic immunity standard for residential, commercial, and light industry

  EN50082–2 Generic immunity standard for industrial environment

- **Low Voltage Directive Standards Applicable to PLCs**

  EN61010–1 Safety requirements for electrical equipment for measurement, control, and laboratory use.

- **Product Specific Standard for PLCs**

  EN61131–2 Programmable controllers, equipment requirements and tests. This standard replaces the above generic standards for immunity and safety. However, the generic emissions standards must still be used in conjunction with the following standards:

  EN 61000-3-2 Harmonics

  EN 61000-3-2 Fluctuations

WARNING: Electrostatic Discharge (ESD)

We recommend that all personnel take necessary precautions to avoid the risk of transferring static charges within the control cabinet and provide clear warnings and instructions on the cabinet exterior. Such precautions may include the use of earth straps, grounding mats and similar static-control devices, or the powering off of the equipment inside the enclosure before the door is opened.

WARNING: Radio Interference (RFI)

This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate preventative measures.

With respect to the D2-262:

The following are product specific standard for CPUs and covers the low voltage and EMC directives as required for European CE certification. This standard has many tests together with test procedures and limits, but also references the below standards for some tests as they apply to the D2-262, manufactured by Koyo Electronics Industries.

IEC 60068	IEC 60417	IEC 60664	IEC 60695	IEC 60707	IEC 60947	IEC 60950	IEC 61000	IEC 61010
-2-1:1990 Part 2 Test A	All Parts	-1:1992 Part 1	-2-1 (all sheets) Part 2	:1999	-5-1:1997 Part 5-1	-1:2001 Part 1	-4-2:1995 Part 4-2	-1:2001 Part 1
-2-2:1974 Part 2 Test B		-3:1992			-7-1:2002 Part 7-1		-4-3:2002 Part 4-3	
-2-6:1995 Part 2: Test Fc							-4-4:1995	
-2-6:1995 Part 2: Test Fc		CISPR 11:1999					-4-5:1995 Part 4-5	
-2-14:1984 Part 2 Test N		CISPR 16-1:1999 Part 1					-4-6:1996 Part 4-6	
-2-27:1987 Part 2 Test Ea		CISPR 16-2:1999 Part 2					-4-8:1993 Part 4-8	
-2-30:1980 Part 2 Test Db							-4-12:1995 Part 4-12	
-2-31:1969 Part 2 Test Ec								
-2-32:1975 Part 2 Test Ed								

For undated references, the latest edition of the referenced document (including any amendments) applies.

General Safety

- External switches, circuit breaker or external fusing are required for these devices.
- The switch or circuit breaker should be mounted near the programmable controller equipment.

As of this printing AutomationDirect is in the process of changing their testing procedures from the generic standards to the product specific standards.

Other Sources of Information

Although the EMC Directive gets the most attention, other basic Directives such as the Machinery Directive and the Low Voltage Directive, also place restrictions on the control panel builder. Because of these additional requirements it is recommended that the following publications be purchased and used as guidelines:

- BSI publication BS TH 42073: November 2000 – covers the safety and electrical aspects of the Machinery Directive
- EN 60204–1:2016 – Safety of Machinery; General electrical requirements for machinery, including Low Voltage and EMC considerations
- IEC 61000–5–2: EMC earthing and cabling requirements
- IEC 61000–5–1: EMC general considerations

It may be possible for you to obtain this information locally; however, the official source of applicable Directives and related standards is:

- **The Office for Official Publications of the European Communities** L–2985 Luxembourg.

The quickest contact is via the World Wide Web at http://publications.europa.eu/index_en.htm

Other sources are:

British Standards Institution – Sales Department

Linford Wood

Milton Keynes

MK14 6LE

United Kingdom;

The quickest contact is via the World Wide Web at http://www.bsi.org.uk

Or a commercial provider of Standards at www.ihs.com

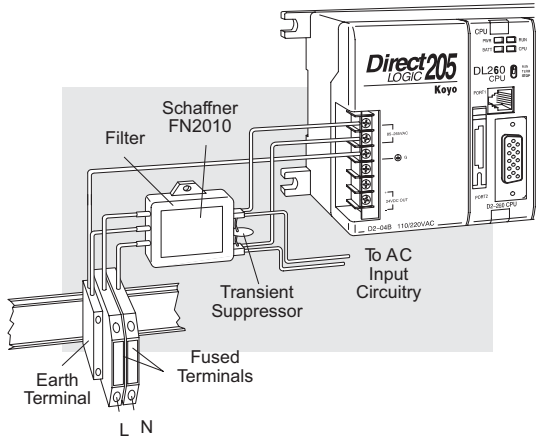
Basic EMC Installation Guidelines

Enclosures

The simplest way to meet the safety requirements of the Machinery and Low Voltage Directives is to house all control equipment in an industry standard lockable steel enclosure. Normally this has an added benefit because it will also help ensure that the EMC characteristics are well within the requirements of the EMC Directive. Although the RF emissions from the PLC equipment, when measured in the open air, are well below the EMC Directive limits, certain configurations can increase emission levels. Enclosure penetrations, for the passage of cables or to mount operator interfaces, will often increase emissions.

AC Mains Filters

DL05, DL06, and DL205 AC powered base power supplies require extra mains filtering to comply with the EMC Directive on conducted RF emissions. All PLC equipment has been tested with filters from Schaffner, which reduce emissions levels when the filters are properly grounded (earth ground). A filter with a current rating suitable to supply all PLC power supplies and AC input modules should be selected. We suggest the FN2010 for DL05/DL06/ DL205 systems.



NOTE: *Very few mains filters can reduce problem emissions to negligible levels. In some cases, filters may increase conducted emissions if not properly matched to the problem emissions.*

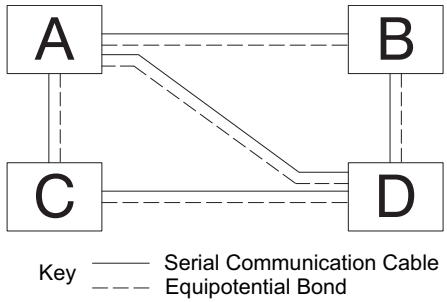
Suppression and Fusing

In order to comply with the fire risk requirements of the Low Voltage and Machinery Directive electrical standards (EN 61010–1 and EN 60204–1), by limiting the power into "unlimited" mains circuits with power leads reversed, it is necessary to fuse both AC and DC supply inputs. You should also install a transient voltage suppressor across the power input connections of the PLC. Choose a suppressor such as a metal oxide varistor with a rating of 275VAC working voltage for 230V nominal supplies (150VAC working voltage for 115V supplies) and high energy capacity (e.g., 140 joules).

Transient suppressors must be protected by fuses, and the capacity of the transient suppressor must be greater than the blow characteristics of the fuses or circuit breakers to avoid a fire risk. A recommended AC supply input arrangement for Koyo PLCs is to use twin 3 amp TT fused terminals with fuse blown indication, such as DINnectors DN–F10L terminals, or twin circuit breakers, wired to a Schaffner FN2010 filter or equivalent, with high energy transient suppressor soldered directly across the output terminals of the filter. PLC system inputs should also be protected from voltage impulses by deriving their power from the same fused, filtered, and surge-suppressed supply.

Internal Enclosure Grounding

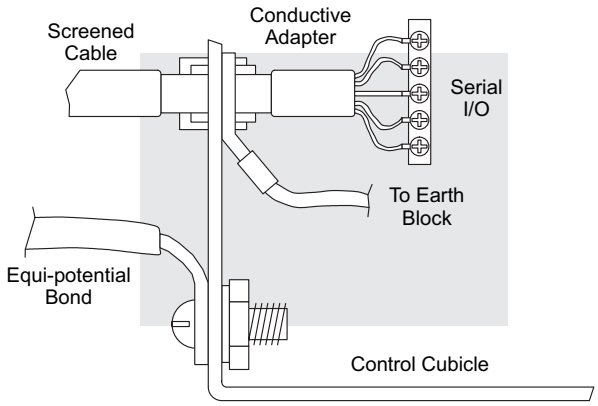
A heavy-duty star earth terminal block should be provided in every cubicle for the connection of all earth ground straps, protective earth ground connections, mains filter earth ground wires, and mechanical assembly earth ground connections. This should be installed to comply with safety and EMC requirements, local standards, and the requirements found in IEC 1000–5–2. The Machinery Directive also requires that the common terminals of PLC input modules, and common supply side of loads driven from PLC output modules, should be connected to the protective earth ground terminal.



Equipotential Grounding

Adequate site earth grounding must be provided for equipment containing modern electronic circuitry. The use of isolated earth electrodes for electronic systems is forbidden in some countries. Make sure you check any requirements for your particular destination. IEC 1000–5–2 covers equipotential bonding of earth grids adequately, but special attention should be given to apparatus and control cubicles that contain I/O devices, remote I/O racks, or have inter-system communications with the primary PLC system enclosure. An equipotential bond wire must be provided alongside all serial communications cables, and to any separate items of the plant which contain I/O devices connected to the PLC. The diagram above shows an example of four physical locations connected by a communications cable.

Communications and Shielded Cables



Good quality 24AWG minimum twisted-pair shielded cables, with overall foil and braid shields, are recommended for analog cabling and communications cabling outside of the PLC enclosure. To date, it has been a common practice to only provide an earth ground for one end of the cable shield in order to minimize the risk of noise caused by earth ground loop currents between apparatus. The procedure of only grounding one end, which primarily originated as a result of trying to reduce hum in audio systems, is no longer applicable to the complex industrial environment. Shielded cables are also efficient emitters of RF noise from the PLC system, and can interact in a parasitic manner in networks and between multiple sources of interference.

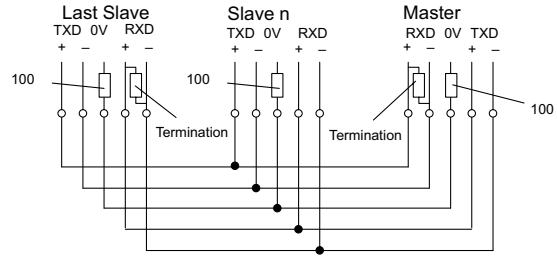
The recommendation is to use shielded cables as electrostatic "pipes" between apparatus and systems, and to run heavy gauge equipotential bond wires alongside all shielded cables. When a shielded cable runs through the metallic wall of an enclosure or machine, it is recommended in IEC 1000–5–2 that the shield should be connected over its full perimeter to the wall, preferably using a conducting adapter, and not via a pigtail wire connection to an earth ground bolt. Shields must be connected to every enclosure wall or machine cover that they pass through.

Analog and RS232 Cables

Providing an earth ground for both ends of the shield for analog circuits provides the perfect electrical environment for the twisted pair cable as the loop consists of signal and return, in a perfectly balanced circuit arrangement, with connection to the common of the input circuitry made at the module terminals. RS232 cables are handled in the same way.

Multi-drop Cables

RS422 twin twisted pair and RS485 single twisted pair cables also require a 0V link, which has often been provided in the past by the cable shield. It is now recommended that you use triple twisted pair cabling for RS422 links, and twin twisted pair cable for RS485 links. This is because the extra pair can be used as the 0V inter-system link. With loop DC power supplies earth grounded in both systems, earth loops are created in this manner via the inter-system 0v link. The installation guides encourage earth loops, which are maintained at a low impedance by using heavy equipotential bond wires. **To account for non–European installations using single-end earth grounds, and sites with far from ideal earth ground characteristics, we recommend the addition of 100 ohm resistors at each 0V link connection in network and communications cables.**



Shielded Cables within Enclosures

When you run cables between PLC items within an enclosure that also contains susceptible electronic equipment from other manufacturers, remember that these cables may be a source of RF emissions. There are ways to minimize this risk. Standard data cables connecting PLCs and/or operator interfaces should be routed well away from other equipment and its associated cabling. You can make special serial cables where the cable shield is connected to the enclosure's earth ground at both ends, the same way that external cables are connected.

Analog Modules and RF Interference

All Automationdirect products are tested to withstand field strength levels up to 10V/m, which is the maximum required by the relevant EU standards. While all products pass this test, analog modules will typically exhibit deviations of their readings. This is quite normal; however, systems designers should be aware of this and plan accordingly.

When assembling a control system using analog modules, these issues must be adhered to and should be integrated into the system design. This is the responsibility of the system builder/ commissioner.

Network Isolation

For safety reasons, it is a specific requirement of the Machinery Directive that a keyswitch must be provided that isolates any network input signal during maintenance, so that remote commands cannot be received that could result in the operation of the machinery. The FA–ISOCON does not have a keyswitch! Use a keylock and switch on your enclosure which, when open, removes power from the FA–ISOCON. To avoid the introduction of noise into

the system, any keyswitch assembly should be housed in its own earth grounded steel box and the integrity of the shielded cable must be maintained.

Again, for further information on EU directives, we recommend that you get a copy of our EU Installation Manual (DA–EU–M). Also, if you are connected to the World Wide Web, you can check the EU Commission's official site at: **http://ec.europa.eu/index_en.htm**.

DC Powered Versions

Due to slightly higher emissions radiated by the DC powered versions of the DL205, and the differing emissions performance for different DC supply voltages, the following stipulations must be met:

- The PLC must be housed within a metallic enclosure with a minimum number of orifices.
- I/O and communications cabling exiting the cabinet must be contained within metallic conduit/trunking.

Items Specific to the DL205

- The rating between all circuits in this product are rated **as basic insulation only**, as appropriate for single fault conditions.
- There is no isolation offered between the PLC and the analog inputs of this product.
- It is the responsibility of the system designer to earth one side of all control and power circuits, and to earth the braid of screened cables.
- This equipment must be properly installed while adhering to the guidelines of the installation manual DA–EU–M (available for download at AutomationDirect Technical Support Manuals), and the installation standards IEC 1000–5–1, IEC 1000–5–2 and IEC 1131–4.
- It is a requirement that all PLC equipment must be housed in a protective steel enclosure, which limits access to operators by a lock and power breaker. If operators or untrained personnel require access, the equipment must be installed inside an internal cover or secondary enclosure. A warning label must be used on the front door of the installation cabinet as follows:

  Warning: Exposed terminals and hazardous voltages inside.

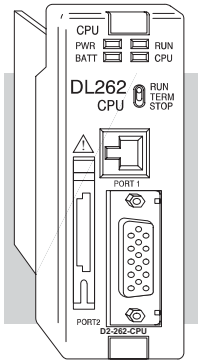
- It should be noted that the safety requirements of the machinery directive standard EN60204–1 state that all equipment power circuits must be wired through isolation transformers or isolating power supplies, and that one side of all AC or DC control circuits must be earthed.
- Both power input connections to the PLC must be separately fused using 3 amp T-type anti–surge fuses, and a transient suppressor fitted to limit supply overvoltages.
- If the user is made aware through the documentation that the equipment, when used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

D2-262 CPU

In This Appendix...

CPU Overview

The D2-262 CPU is an addition to the DL205 system. It shares the same features as the D2-260 CPU. This appendix provides the information needed to understand the differences between the D2-262 and D2-260 CPUs.



General CPU Features

The D2–262 is a modular CPU in the same family as D2-230, D2-240, D2-250–1 and D2–260 CPUs. It can be installed in 3, 4, 6, or 9 slot bases. All I/O modules (except the D2-CTRINT) in the DL205 family will work with this CPU. The D2-262 CPU offers a wide range of processing power and program instructions. It offers RLL and Stage program instructions (See Chapter 5). It also provides extensive internal diagnostics that can be monitored from the application program or from an operator interface.

D2-262 CPU Features

The D2-262 offers all the features of the D2-260; it also supports up to 1280 local I/O points by using up to four local expansion bases. It has a maximum of 30.4K of program memory comprised of 15.8K of ladder memory (saved on flash memory) and 14.6K of V-memory (data registers). It also includes an internal RISC–based microprocessor for greater processing power. The D2-262 has 297 instructions. The D2-262 instruction set includes table instructions, trigonometric instructions and support for 16 PID loops.

The D2-262 has two built–in communications ports. The ports are identical to the ports of the D2-260. The ports will interface with ***Direct***SOFT operator interfaces, and provides ***Direct***Net, Modbus RTU Master/Slave connections.

D2-262 CPU Environmental Specifications

D2-262 CPU Environmental Specifications	
Ambient Operating Temperature	32°F to 131°F (0°C to 55°C)
Storage Temperature	-4°F to 158°F (-20°C to 70°C)
Operating Humidity	30% to 95% (Non-condensing)
Atmosphere	No corrosive gases, The level for the environmental pollution is 2 (UL 840)
Vibration Resistance	IEC60068-2-6
Shock Resistance	IEC60068-2-27
Noise Immunity	Impulse noise µs, 1000V RFI (145MHz, 435MHz)
Agency Approval	UL, CE, (FCC CLASS A), Class 1 Division 2 (C1D2)
Internal Power Consumption	DC 5V, 336mA Max.
Acceptable External Power Drop	Max. 10ms, (It Require more than 900ms interval between on and off)
Size	1.20" W x 3.50" H x 2.65" D (30.8 W x 89.9 H x 68 D mm)
Weight	2.5 oz. (70g)

DL205 CPU Bases Electrical Specifications

Specification	AC Powered Base	24 VDC Powered Base	125 VDC Powered Base
Part Numbers	D2–03B–1 D2–04B–1 D2–06B–1 D2–09B–1	D2–03BDC1–1 D2–04BDC1–1 D2–06BDC1–1 D2–09BDC1–1	D2–06BDC2–1 D2–09BDC2–1
Input Voltage Range	100–240 VAC +10% –15%	10.2–28.8 VDC (24VDC) with less than 10% ripple	104–240 VDC +10% –15%
Maximum Inrush Current	30A	10A	20A
Maximum Power	80VA	25W	30W
Voltage Withstand (dielectric)	1 minute @ 1500VAC between primary, secondary, field ground, and run relay		
Insulation Resistance	> 10MΩ at 500VDC		
Auxiliary 24 VDC Output	20–28 VDC, less than 1V p-p 300mA max.	None	20–28 VDC, less than 1V p-p 300mA max.
Fusing (internal to base power supply)	Non–replaceable 2A @ 250V slow blow fuse	Non–replaceable 3.15 A @ 250V slow blow fuse	Non–replaceable 2A @ 250V slow blow fuse

D2-260/D2-262 CPU General Specifications

Feature	D2-260	D2-262
Total Program memory (words)	30.4K	
Ladder memory (words)	15872 (Flash)	
V-memory (words)	14592	
Non-volatile V Memory (words)	No	
Boolean execution /K	1.9 ms/1.0 ms	
RLL and RLL^{PLUS} Programming	Yes	
Handheld programmer	Yes	
*Direct*SOFT programming for Windows.	Yes (requires version 4.0 or higher)	Yes (requires version 6.3 or higher)
Built-in communication ports	One RS–232 One RS–232, RS–422 or RS–485	
EEPROM	Flash	
Total CPU memory I/O points available	8192 (X, Y, CR, GX, GY)	
Local I/O points available	256	
Local Expansion I/O points (including local I/O and expansion I/O points)	1280 (4 exp. bases max.)	
Serial Remote I/O points (including local I/O and expansion I/O points)	8192	
Serial Remote I/O Channels	8	
Max Number of Serial Remote Slaves	7 Remote / 31 Slice	
Ethernet Remote I/O Discrete points	8192	
Ethernet Remote I/O Analog I/O channels	Map into V–memory	
Ethernet Remote I/O channels	Limited by power budget	
Max Number of Ethernet slaves per channel	16	
I/O points per Remote channel	16,384 (16 fully expanded H4–EBC slaves using V–memory and bit–of–word instructions	
I/O Module Point Density	4/8/12/16/32	
Slots per Base	3/4/6/9	

D2-262 CPU General Specifications

Feature	D2-260 or D2-262
Number of instructions available (see Chapter 5 for details)	297
Control relays	2048
Special relays (system defined)	144
Stages in RLL^{PLUS}	1024
Timers	256
Counters	256
Immediate I/O	Yes
Interrupt input (hardware / timed)	Yes / Yes
Subroutines	Yes
Drum Timers	Yes
Table Instructions	Yes
For/Next Loops	Yes
Math	Integer, Floating Point, Trigonometric
ASCII	Yes, IN/OUT
PID Loop Control, Built In	Yes, 16 Loops
Time of Day Clock/Calendar	Yes
Run Time Edits	Yes
Supports Overrides	Yes
Internal diagnostics	Yes
Password security	Yes
System error log	Yes
User error log	Yes
Battery backup	Yes (optional)

D2-260/D2-262 CPU Program/Memory Specifications

Items		Specification	
Control Method		Stored program / Cyclic execution method	
I/O Transfer Method		3 Methods: In a lump, Direct (by instructions), Fixed interval (by the fixed scan)	
Program Language		Ladder Logic (Standard RLL), Stage logic (RLLPLUS)	
		CPUs	
		D2-260	D2-262
Number of Instructions		297	
Execution Time		Standard Instruction 0.61 µs ~ Advanced Instruction 2.0 µs ~ Scan time 1.9 ms/ K words	Standard Instruction 0.1 µs ~ Advanced Instruction 0.1 µs~ Scan time 1ms / K words
Operand Numbering System		Octal system (this numbering system depends on peripheral equipments.)	
Ladder Memory Size		15872 Words	
Number of I/O points	I/O	1024 Inputs (X0–X1777)	
		1024 Outputs (Y0–Y1777)	
	Remote I/O	Yes, (GX,GY,X,Y,C,V)	
Link Inputs		2048 (GX0–GX3777)	
Link Output		2048 (GY0–GY3777)	
Control Relays		2048 (C0–C3777)	
Timers		256 (T0–T 377)	
Counters		256 (CT0–CT377)	
Stages		1024 (S0–S1777)	
V-memory		14592 words (V400–V777, V1400–V7377,V10000–35777)	
System V-memory		1152 words (V7600–V7777, V36000–37777)	
Special Relays		512 (SP0–SP777), 144 (Used)	
Accumulator		32 bits x 1	
Accumulator Stack		32 bits x 8	
Clock / Calendar		Year, Month, Day, Day of week, Hour, Min, Sec, 1/100 sec: in system V-memory	
I/O Numbering System		Fixed.(Input 0–1777, Output Y0–1777); Variable (by system of optional I/O & expansion I/O)	
PID Loop Operation		16 Loop	
Memory Backup		A Super Capacitor	Data: SRAM Program: Flash ROM
Software Interrupt		1 (5–1000 ms)	

D2-260/D2-262 CPU Program/Memory Specifications

Users Memory

Users Memory	Range	D2-260	D2-262
Total Memory available (words)		30.1K Words	
L Memory (words)	0 - 14591	15872 Words	
V Memory (words) *Data Register	V 400 – 777 V 1400-7377 V10000-35777	14592 Words	

Bit map - D2-260/ D2-262

Memory Type	Discrete Memory Reference	QTY (Bits)	Word Memory Reference	Qty (words)
Link Input Points	GX 0–3777	2048	V 40000–40177	128
Link Output Points	GY 0–3777	2048	V 40200–40377	128
Input Points	X 0–1777	1024	V 40400–40477	64
Output Points	Y 0–1777	1024	V 40500–40577	64
Control Relays	C 0–3777	2048	V 40600–40777	128
Stages	S 0–1777	1024	V 41000–41077	64
Timer Status	T 0–377	256	V 41100–41117	16
Counter Status	CT 0–377	256	V 41140–41157	16
Special Relays	SP 0–777	512	V 41200–41237	32

V-Memory Map - D2-260/ D2-262

V-Memory Type	V-Memory Reference	Qty (words)
Timer Current Values	V 0 – 377	256
Data Words	V 400 - 777	256
Counter Current Values	V 1000 – 1377	256
Data Words	V 1400 - 7377	3072
System Parameters	V 7400 - 7777	256
Data Words	V 10000 - 35777	11264
Special Data Words	V 36000 - 37777	1024

Expansion Modules Supported by D2-260 and D2-262

Analog Module	D2-260	D2-262
F2-04AD-1	√	√
F2-04AD-2	√	√
F2-08AD-1	√	√
F2-08AD-2	√	√
F2-02DA-1	√	√
F2-02DA-2	√	√
F2-02DA-1L	√	√
F2-02DA-2L	√	√
F2-02DAS-1	√	√
F2-02DAS-2	√	√
F2-08DA-1	√	√
F2-08DA-2	√	√
F2-4AD2DA	√	√
F2-04RTD	√	√
F2-04THM	√	√
F2-8AD4DA-1	√	√
F2-8AD4DA-2	√	√

Intelligent Module *	D2-260	D2-262
D2-RMSM	√	√
H2-ECOM	√	√
H2-ECOM100	√	√
D2-DCM	√	√
H2-CTRIO	√	√
D2-CTRINT	√	Not supported
D2-CM	√	√
D2-EM	√	√
H2-ERM100	√	√
H2-EBC100	√	√
F2-DEVNETS-1	√	√
F2-SDS-1	√	√
F2-CP128	√	√
H2-CTRIO2	√	√
H2-SERIO	Not supported	Not supported
H2-SERIO-4		

*** Intelligent I/O cannot be used in an expansion base.**

Discrete Module	D2-260	D2-262
F2-08SIM	√	√
D2-08ND3	√	√
D2-16ND3-2	√	√
D2-32ND3	√	√
D2-32ND3-2	√	√
D2-08NA-1	√	√
D2-08NA-2	√	√
D2-16NA	√	√
D2-04TD1	√	√
D2-08TD1	√	√
D2-08TD2	√	√
D2-16TD1-2	√	√
D2-16TD2-2	√	√
D2-32TD1	√	√
D2-32TD2	√	√
D2-08TA	√	√
F2-08TA	√	√
D2-12TA	√	√
D2-04TRS	√	√
D2-08TR	√	√
F2-08TR	√	√
F2-08TRS	√	√
F2-16D1P	√	√
F2-16D2P	√	√
D2-12TR	√	√
D2-08CDR	√	√