AUTOMATIONDIRECT.com

D4-454 PLC User Manual

Manual Number: D4-454-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.

Nulle partie de ce manuel ne doit être copiée, reproduite ou transmise de quelque façon que ce soit sans le consentement préalable écrit de la société **Automationdirect.com® Incorporated**. **AutomationDirect** conserve les droits exclusifs à l'égard de tous les renseignements contenus dans le présent document.

D4-454 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:	D4-454-M
Issue:	1st Ed. Rev. E
Issue Date:	09/23

Publication History

Issue	Date	Description of Changes
Original	06/17	Original Issue
Rev A	09/17	Minor Changes
Rev B	05/18	Updates and general corrections
Rev C	06/20	Added note to Appendix F: D4-454 will only support D4-HSC firmware v2.3.
Rev D	03/23	Removed Type 1 fonts
Rev E	09/23	Updated D4-454 CPU specs - Does not support D4-HSC modules

Table of Contents

Chapter 3: CPU Specifications and Operation

Chapter 5: Standard RLL Instructions

Chapter 6: Drum Instruction Programming

Chapter 7: RLL^{PLUS} Stage Programming

Chapter 9: Maintenance and Troubleshooting

GETTING STARTED

CHAPTER 1

In This Chapter...

Introduction

The Purpose of this Manual

Thank you for purchasing a DL405 PLC. This manual shows you how to install, program, and maintain a D4-454 CPU. It also helps you understand how to interface them to other devices in a control system. This manual contains important information for both installing and programming PLC systems using the D4-454 CPU. This user manual will provide the information you need to get 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. Be sure to keep this manual handy for reference when you have questions. If you are new to the D4-454 PLC controller, we suggest you read this manual completely so you can understand the wide variety of features available. We believe you will be pleasantly surprised with how much you can accomplish with our 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 let us know.

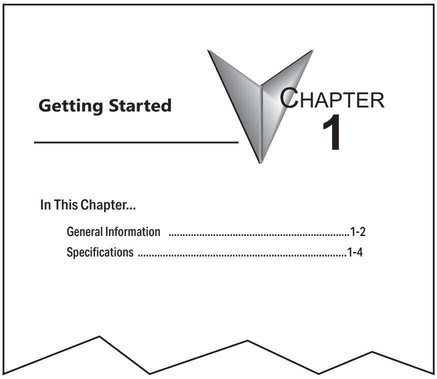
Conventions Used

***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 point 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. Any warning in this manual should be regarded as critical information that should be read in its entirety.
The word WARNING in boldface type 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.



Getting Started

CHAPTER
1

In This Chapter...

General Information 1-2
Specifications 1-4

DL405 System Components

CPUs

There are two feature-enhanced CPUs in this product line, the D4-454 and D4-454DC-1. Both include a built-in power supply and communication ports. Each CPU offers a large amount of program memory, a substantial instruction set and advanced diagnostics. The D4-454 CPUs features drum timers, floating-point math, built-in PID loops, and additional communications ports. Details of these CPU features are covered in Chapter 3, CPU Specifications and Operation.

Bases

Three base sizes are available in the system: 4 slot, 6 slot and 8 slot.

I/O Configuration

The D4-454 can support up to 1024 local I/O. A maximum of 1536 I/O points for the D4-454 can be added to the system in the form of remote I/O bases and slice I/O blocks. Each of these I/O configurations is explained in Chapter 4, System Design and configuration.

I/O Modules

The DL405 family has a complete range of discrete modules which support 24 VDC, 125 VDC, 110/220 VAC and up to 10A relay outputs. Analog modules provide 12-bit resolution and several selections of input and output signal ranges (including bipolar). Specialty modules include high-speed inputs and outputs, thermocouple, general purpose communication, magnetic pulse input, 16 loop PID function and more.

Programming Methods

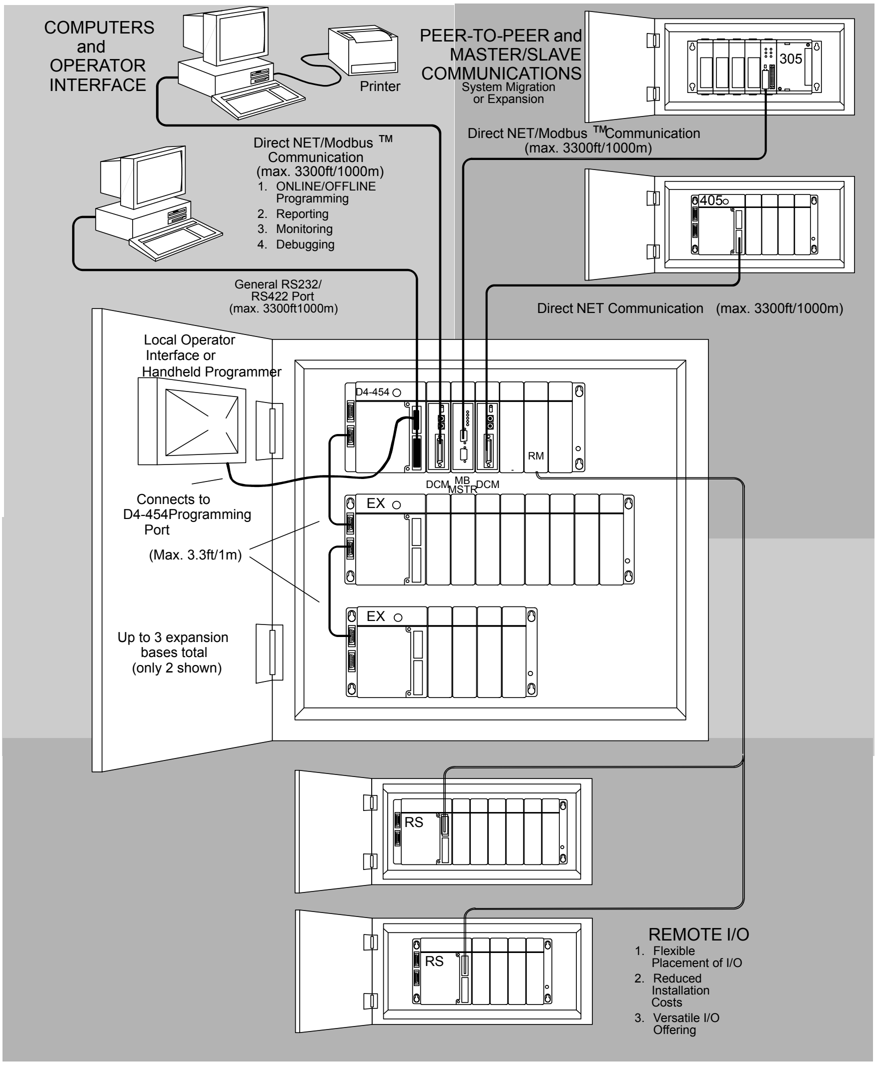
DirectSOFT Programming for Windows™

There are two programming methods available for the D4-454 CPUs, RLL (Relay Ladder Logic) and RLLPLUS Stage Programing. Both the DirectSOFT programming package and the handheld programmer support RLL and stage.

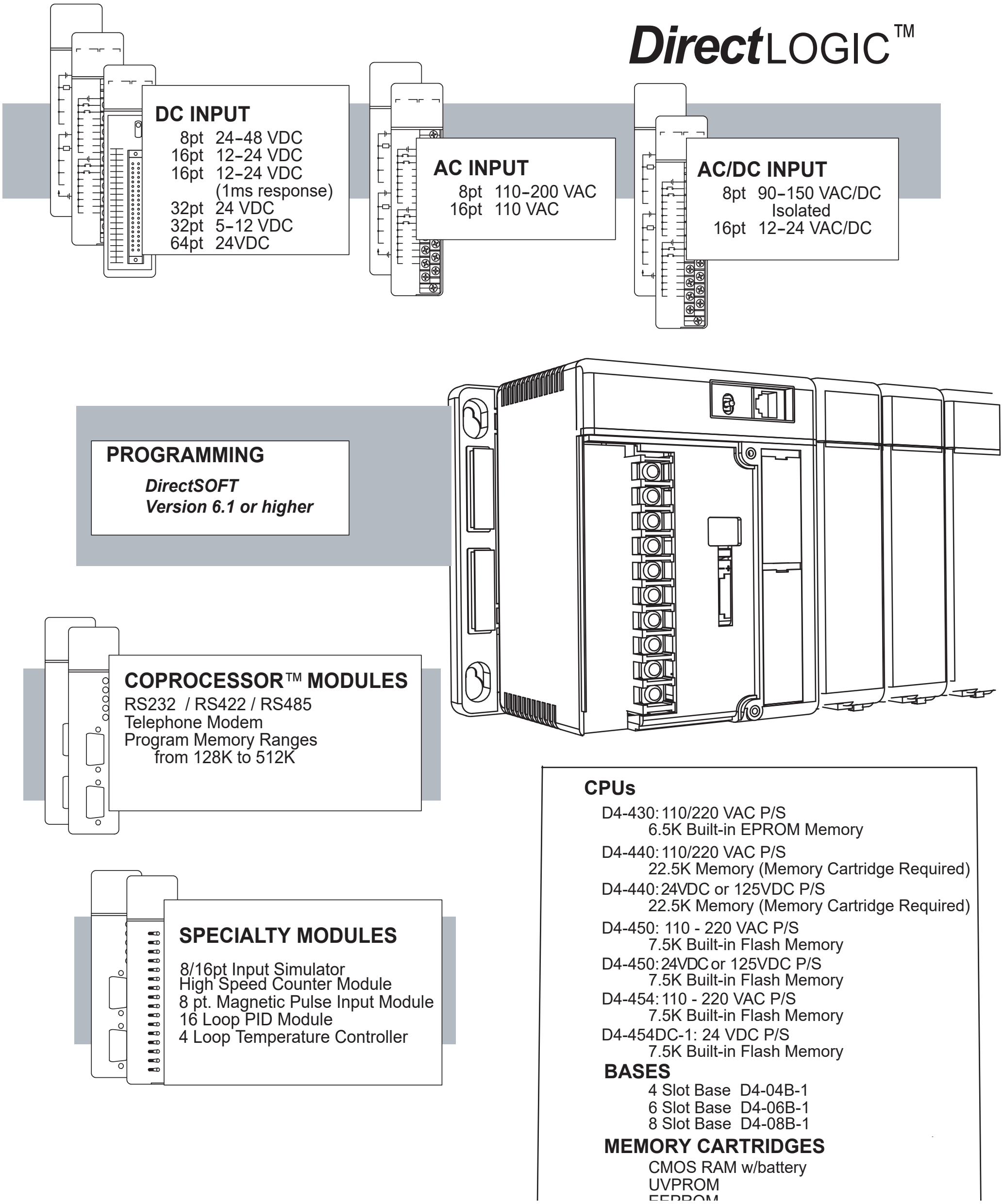
The D4-454 CPUs can be programmed with DirectSOFT Version 6.1 or later. DirectSOFT is a Windows-based software package that supports many of the Windows features you already know, such as cut-and-paste between applications, point-and-click editing, viewing and editing multiple application programs at the same time, browsers, and Intelligent Box (IBox) Instructions which will help ease your programming tasks. DirectSOFT universally supports the DirectLOGIC™ CPU families. This means you can use the same DirectSOFT package to program DL05 ,DL06, DL105, DL205, DL305, DL405 or any new DirectLOGIC CPUs that we add to our product line. There is a separate manual for the DirectSOFT programming software and 1 copy is shipped with each purchase.

DL405 System Diagrams

The diagram below shows the major components and configurations of the DL405 system. The next two pages show specific components for building your system.


COMPUTERS and OPERATOR INTERFACE
Printer
PEER-TO-PEER and MASTER/SLAVE COMMUNICATIONS
System Migration or Expansion
305
Direct NET/Modbus TM Communication (max. 3300ft/1000m)
Direct NET/Modbus TM Communication (max. 3300ft/1000m)
1. ONLINE/OFFLINE Programming
2. Reporting
3. Monitoring
4. Debugging
405
General RS232/ RS422 Port (max. 3300ft1000m)
Direct NET Communication (max. 3300ft/1000m)
Local Operator Interface or Handheld Programmer
D4-454
RM
DCM
MB MSTR
DCM
EX
Connects to D4-454Programming Port
(Max. 3.3ft/1m)
EX
Up to 3 expansion bases total (only 2 shown)
RS
RS
REMOTE I/O
1. Flexible Placement of I/O
2. Reduced Installation Costs
3. Versatile I/O Offering

DL405 Family



*Direct*LOGIC™

DC INPUT

8pt 24–48 VDC
16pt 12–24 VDC
16pt 12–24 VDC
(1ms response)
32pt 24 VDC
32pt 5–12 VDC
64pt 24VDC

AC INPUT

8pt 110–200 VAC
16pt 110 VAC

AC/DC INPUT

8pt 90–150 VAC/DC
Isolated
16pt 12–24 VAC/DC

PROGRAMMING

DirectSOFT
Version 6.1 or higher

COPROCESSOR™ MODULES

RS232 / RS422 / RS485
Telephone Modem
Program Memory Ranges
from 128K to 512K

SPECIALTY MODULES

8/16pt Input Simulator
High Speed Counter Module
8 pt. Magnetic Pulse Input Module
16 Loop PID Module
4 Loop Temperature Controller

CPUs

D4-430: 110/220 VAC P/S
6.5K Built-in EPROM Memory
D4-440: 110/220 VAC P/S
22.5K Memory (Memory Cartridge Required)
D4-440: 24VDC or 125VDC P/S
22.5K Memory (Memory Cartridge Required)
D4-450: 110 - 220 VAC P/S
7.5K Built-in Flash Memory
D4-450: 24VDC or 125VDC P/S
7.5K Built-in Flash Memory
D4-454: 110 - 220 VAC P/S
7.5K Built-in Flash Memory
D4-454DC-1: 24 VDC P/S
7.5K Built-in Flash Memory

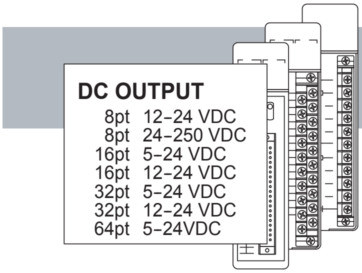
BASES

4 Slot Base D4-04B-1
6 Slot Base D4-06B-1
8 Slot Base D4-08B-1

MEMORY CARTRIDGES

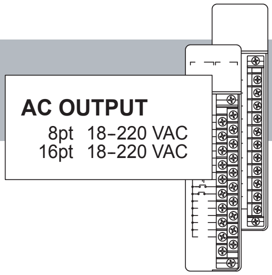
CMOS RAM w/battery
UVPROM
EEPROM

DL405 Family



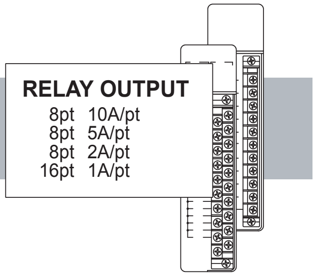
DC OUTPUT

8pt 12–24 VDC
8pt 24–250 VDC
16pt 5–24 VDC
16pt 12–24 VDC
32pt 5–24 VDC
32pt 12–24 VDC
64pt 5–24VDC



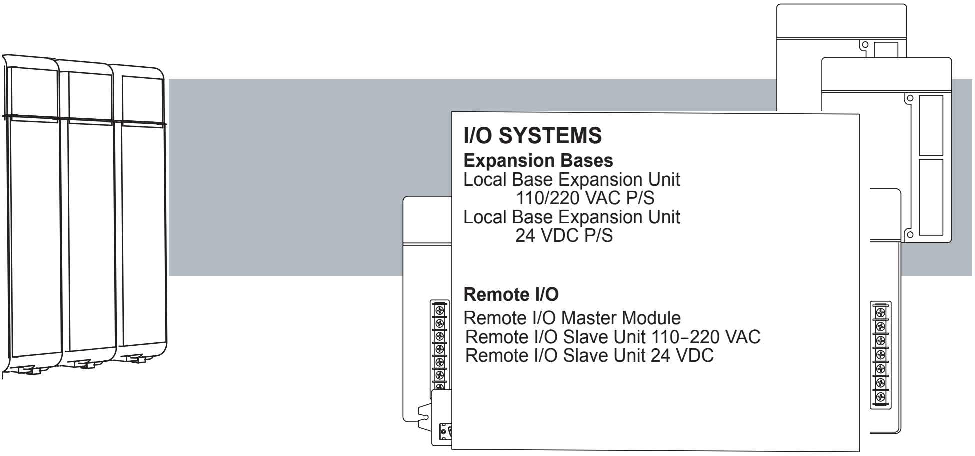
AC OUTPUT

8pt 18–220 VAC
16pt 18–220 VAC



RELAY OUTPUT

8pt 10A/pt
8pt 5A/pt
8pt 2A/pt
16pt 1A/pt



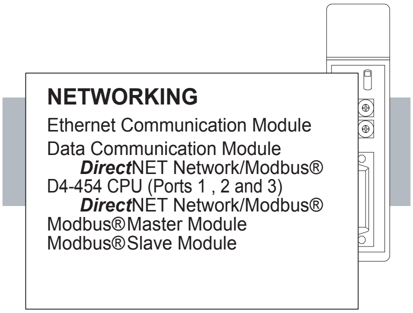
I/O SYSTEMS

Expansion Bases

Local Base Expansion Unit
110/220 VAC P/S
Local Base Expansion Unit
24 VDC P/S

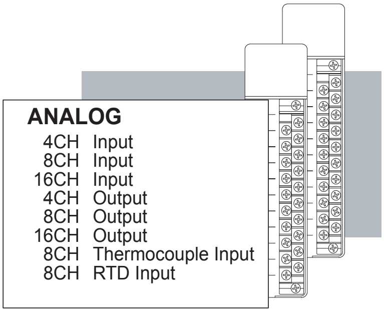
Remote I/O

Remote I/O Master Module
Remote I/O Slave Unit 110–220 VAC
Remote I/O Slave Unit 24 VDC



NETWORKING

Ethernet Communication Module
Data Communication Module
DirectNET Network/Modbus®
D4-454 CPU (Ports 1 , 2 and 3)
DirectNET Network/Modbus®
Modbus® Master Module
Modbus® Slave Module



ANALOG

4CH Input
8CH Input
16CH Input
4CH Output
8CH Output
16CH Output
8CH Thermocouple Input
8CH RTD Input

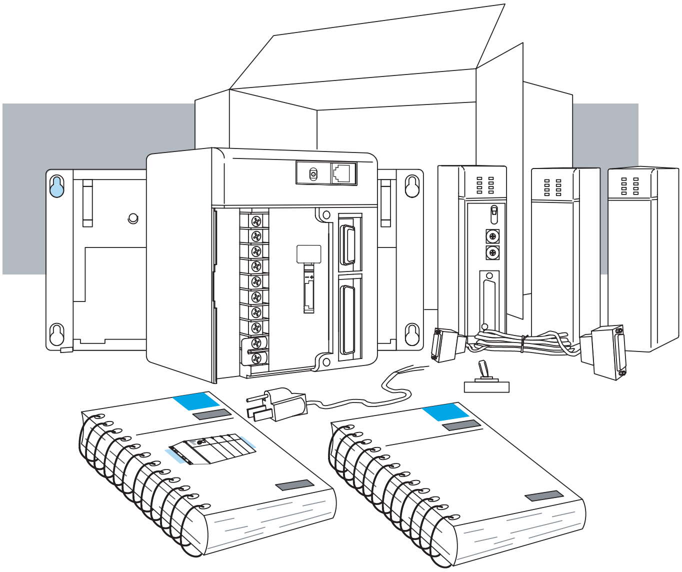
Quick Start for PLC Checkout and Programming

If you have experience with PLCs, or if you just want to setup a quick example, this section is for you! This example is not intended to tell you everything you need to start up your system. It is only intended to give you a general picture of what you will need to do to get your system powered up.

Step 1: Unpack the DL405 Equipment

Unpack the DL405 equipment and verify you have the parts necessary to build this demonstration system. The minimum parts you will need are:

- Base
- CPU
- D4-16ND2 DC input module or a D4-16SIM input simulator module
- D4-16TR
- Power Cord
- Hook up wire
- A 24 VDC toggle switch (if not using the input simulator module)
- A screwdriver, regular or Phillips type
- DirectSOFT Programming Software, DirectSOFT manual, and a programming cable (connects the CPU to a personal computer).



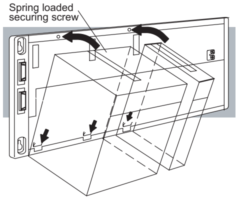
Step 2: Install the CPU and I/O Modules

Insert the CPU and I/O into the base. The CPU must go into the far left side of the base in the position marked “CPU/Power Supply”. When inserting components into the base, tilt the component slightly forward, sliding the tab on the bottom of the component into the slot in the base. Push the top of the component into the base until it is seated firmly, then tighten the securing screw at the top of the module/unit.

WARNING: To minimize the potential damage to the CPU/Module rear base connector(s), do not force the module into position. It should glide smoothly into place. If it does not glide smoothly into place, please check the following: 1) Visually inspect the base connector to make sure the mating holes are not blocked. 2) Visually inspect the pins on the rear connector making sure they are not bent or pushed in. 3) Make sure the plastic tabs on the bottom of the CPU/Module are level before tilting the module into position. Damage to the connectors may occur if the above information is not considered.

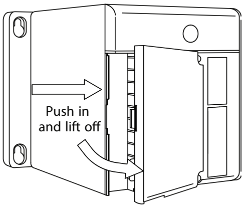
Placement of discrete, analog and relay modules are not critical and may go in any slot in any base, however, for this example install the output module in the slot next to the CPU (slot 0) and the input module in the next slot (slot 1). Limiting factors for other types of modules are discussed in Chapter 4, Bases, Expansion Units and I/O Configuration. You must also make sure you do not exceed the power budget for each base in your system configuration. Power budgeting is discussed in Chapter 4, System Design and Configuration.

- Each unit has a plastic tab at the bottom and screw at the top
- With the unit tilted slightly forward, hook the module’s plastic tab on the base.
- Gently push the top of the unit back until it is firmly installed in the base.
- Secure the unit to the base by tightening the top screw.



Step 3: Remove Terminal Strip Access Cover

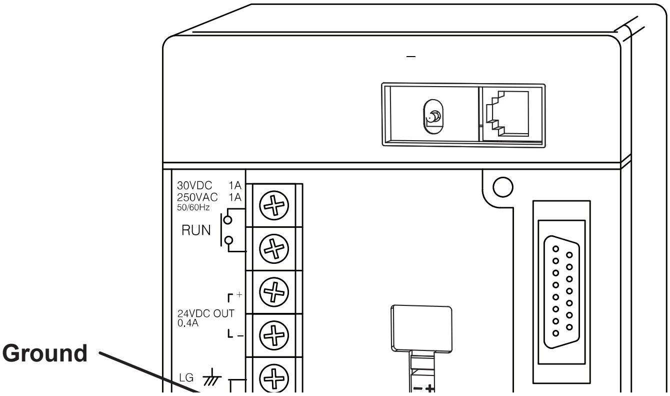
- The terminal strip cover has a small retention cap on the left edge. Push in and up, then pull the cover off.



Step 4: Select Operating Power Range

If you are using 110 VAC, install the voltage select jumper on the bottom two terminals. If you are using 220 VAC power, do not install the jumper. You can find a detailed explanation of the terminal block on both the CPU and expansion units in Chapter 2, Installation, Wiring and Specifications.

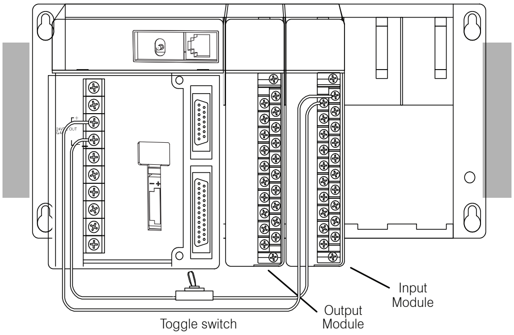
WARNING: Damage will occur to the power supply if 220 VAC is connected to the terminal connections with the 110 VAC jumper in place.



NOTE: *The D4-454 only supports the -1 bases (D4-0xB-1). The D4-454 does not support the non -1 bases (D4-0xB).*

Step 5: Add I/O Simulation

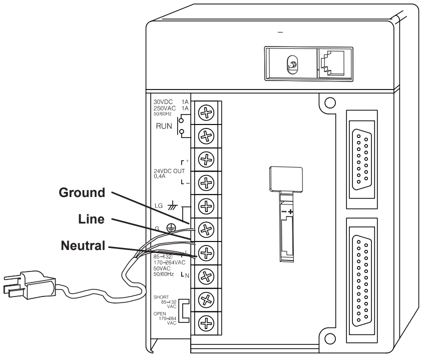
To finish this quick-start exercise or study other examples in the this manual, you'll need to install an input simulator module (or wire an input switch as shown below), and add an output module. Using an input simulator is the quickest way to get physical inputs for checking out the system or a new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



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. Wire the input modules (X0) to the toggle switch and 24 VDC auxiliary power supply on the CPU terminal strip as shown below for the D4-16ND2 input module. Chapter 2, Installation, Wiring and Specifications provides a list of I/O wiring guidelines.

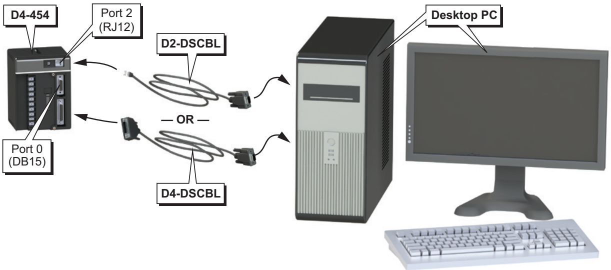
Step 6: Connect the Power Wiring

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



Step 7: Connect Programming Device

Most programmers will use DirectSOFT programming software, installed in a personal computer, to program and monitor the PLC. There are three ports on the D4-454 that can be used for programming cables available from AutomationDirect. For port 0, the 15-pin port, you can use the D4-DSCBL cable. For port 2, the RJ12 port, you can use the D2-DSCBL cable. For port 1, the top portion of the 25-pin port, you can use the D3-DSCBL-2 cable. Also available is the FA-CABKIT. This is a universal kit that allows building a programming cable for all DirectLOGIC PLCs.



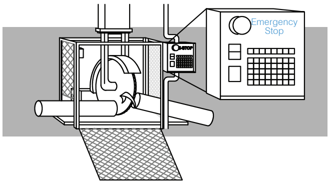
Step 8: Enter the Program

Put the CPUs mode switch in the STOP position to ensure that the CPU is not running a program. Next, put the CPU mode switch in the TERM position. This puts the CPU in the program mode and allows access to the CPU program.

Steps to Designing a Successful System

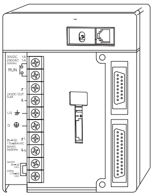
1: Review the Installation Guidelines

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.



2: Understand the CPU Setup Procedures

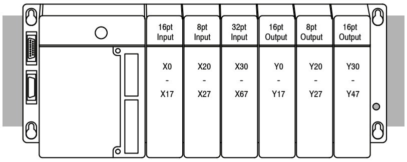
The CPU is the heart of your automation system. Make sure you take time to understand its various features and setup requirements.



3: Understand the I/O System Configurations

It is important to understand how to configure the I/O system. You have several different types of systems:

- Local System
- Expansion System
- Remote I/O System
- Network Connections



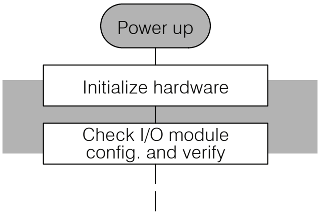
It is also important to understand how the system Power Budget is calculated.

See Chapter 4 for more information.

4: Determine the I/O Module Specifications and Wiring

There are many different I/O modules available with the DL405 system. Chapter 2 provides the specifications and wiring diagrams for the discrete I/O modules.

5: Understand the CPU Operation

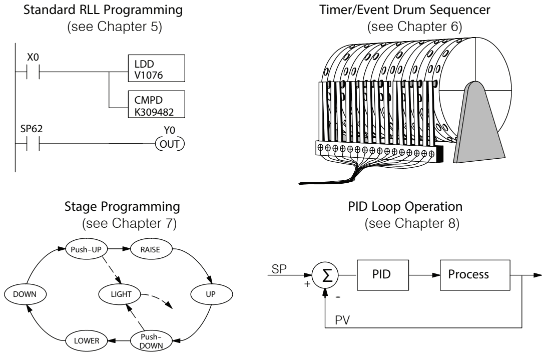


Before you begin to enter a program, it is very helpful to understand how the D4-454 CPU processes information. This involves not only program execution steps, but also involves the various modes of operation and memory layout characteristics. See Chapter 3 for more information.

6: Review the Programming Concepts

The DL405 PLC 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 D4-454 CPUs 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 repetitive process based on a single series of steps.
- Stage programming (also 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 you draw for your process.
- The DL405 PID Loop Operation uses setup tables to configure up to 16 loops. Features include alarms, SP ramp/soak generation and more.

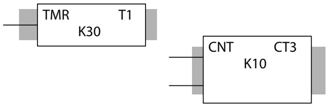

Standard RLL Programming (see Chapter 5)

Timer/Event Drum Sequencer (see Chapter 6)

Stage Programming (see Chapter 7)

PID Loop Operation (see Chapter 8)

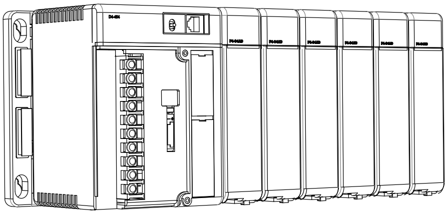
7: Choose the Instructions



After installation and studying the main programming concepts, you can begin writing the application program or configuring loop operation. You'll discover a powerful instruction set.

8: Understanding the Maintenance and Troubleshooting Procedures

Sometimes equipment failures occur when we least expect it. Switches fail, loads short and need replacement, etc. Often the majority of the troubleshooting time is spent in locating the problem. The DL405 system has many built-in features such as error codes that can help you quickly identify problems. See Chapter 9 for diagnostics and troubleshooting tips.



Frequently Asked Questions

Q. How do I reset my D4-454 CPU back to Factory Defaults?

A. Resetting the processor to factory defaults is a two step process. First clear the processor memory using DirectSOFT PLC>CLEAR PL. Next initialize the scratchpad PLC>SETUP>INITIALIZE SCRATCHPAD. Be aware that initializing the scratchpad will reset the system V-memory to defaults. System V-memory contains parameters such as retentive ranges, communication port settings, etc.

Q. How often should the CPU backup battery be changed?

A. Both of the D4-454 CPUs have an LED indicator that will flash when the battery voltage is getting low. The typical battery life is 5 years.

Q. Where can I obtain the most current firmware for the D4-454?

A. In the tech support section of **www.automationdirect.com.** The firmware and instructions on how to update the CPU are available.

Q. Do the DL405 PLCs have Ethernet capability?

A. Yes, the H4-ECOM100 module is needed to support Ethernet.

Q. Are more FAQs available for the DL405 and other products.

A. Yes, visit **www.automationdirect.com** for more FAQs and other technical information.

INSTALLATION, WIRING AND SPECIFICATIONS

CHAPTER 2

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.

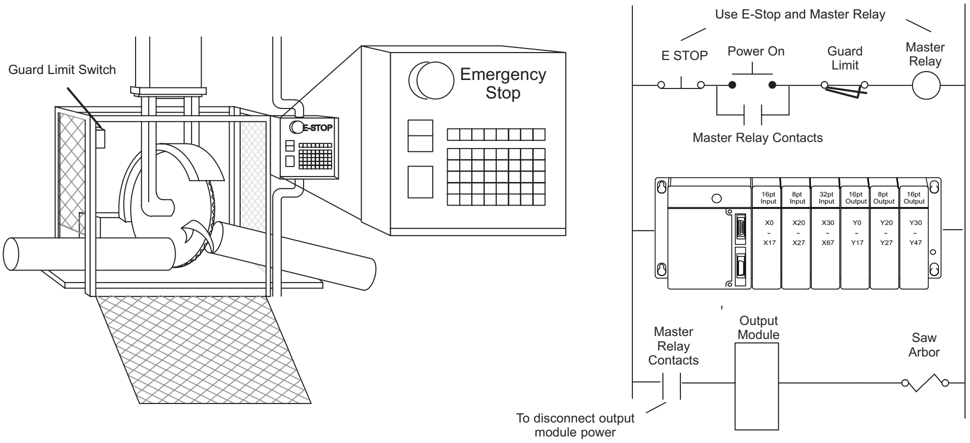
- Emergency stop switch for disconnecting system power
- Mechanical disconnect for output module power
- Orderly system shutdown sequence in the PLC control program

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.).



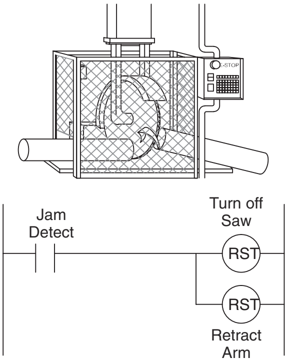
Emergency Power Disconnect

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 “out rush”. 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 insure a known starting point.

Orderly System Shutdown

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



WARNING 1: The control program must not be the only form of protection for any problems that may result in a risk of personal injury or equipment damage.
WARNING 2: This equipment is designed for use in Pollution Degree 2 environments (installed within an enclosure rated at least IP54).
WARNING 3: Transient suppression must be provided to prevent the rated voltage from being exceeded by 140%.

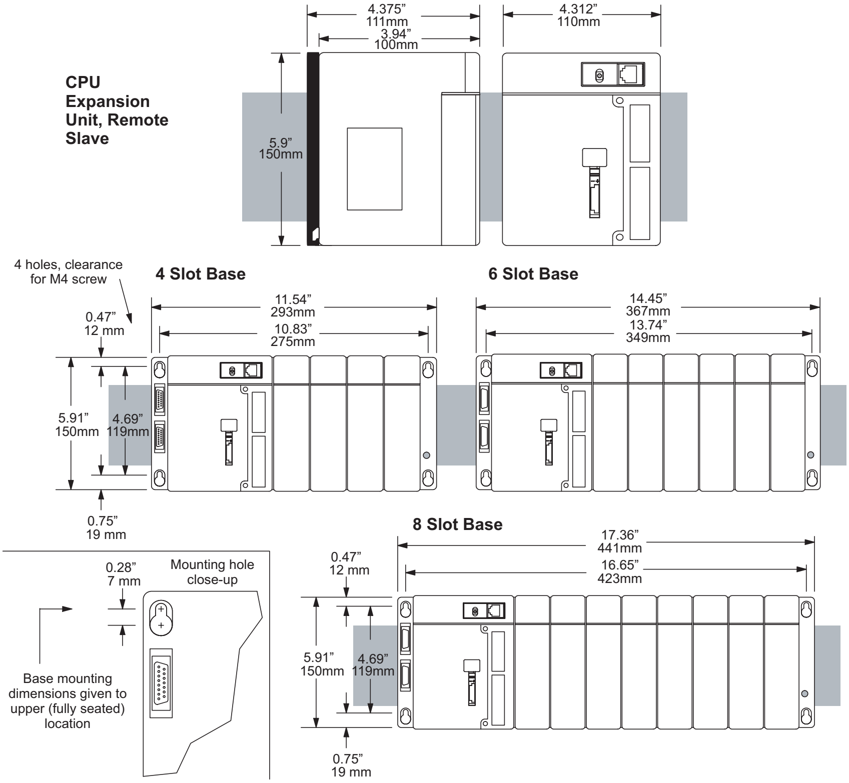
Mounting Guidelines

In addition to the panel layout guidelines, other specifications can affect the installation of a PLC system. Always consider the following:

- Environmental specifications
- Power Supply specifications
- Regulatory Agency Approvals
- Enclosure Selection and Component Dimensions.

Base Dimensions

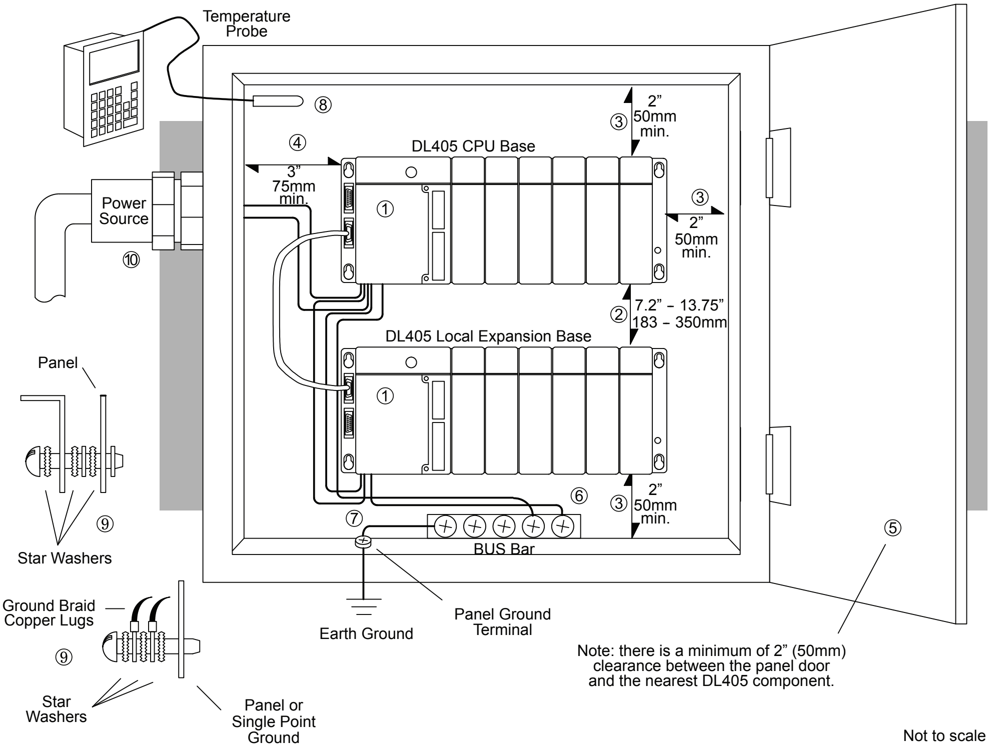
The following diagram shows the outside dimensions and mounting hole locations for the 4-slot, 6-slot, and 8-slot bases. Make sure you follow the installation guidelines to allow proper spacing from other components.



Panel Layout and Clearances

There are many things to consider when designing the panel layout. The following items correspond to the diagram shown on the following page. Note, there may be additional requirements, depending on your application and use of other components in the cabinet.

- The bases must be mounted horizontally to provide proper ventilation.
- There should be a minimum of 7.2” (183mm) and a maximum of 13.75” (350mm) between bases.
- A minimum clearance of 2” (50mm) between the base and the top, bottom and right side of the cabinet should be provided.
- A minimum clearance of 3” (75mm) between the base and the left side of the cabinet should be provided.
- There must be a minimum of 2” clearance between the panel door and the nearest DL405 component.
- The ground terminal on the DL405 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. A general rule is to achieve a 0.1 ohm of DC resistance between the DL405 base and the single point ground.
- 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 as 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 DL405. There are several methods of providing an adequate common ground reference, including:
  a. Installing a ground rod as close to the panel as possible.
  b. Connection to incoming power system ground.
- 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 DL405 system, measures such as installing a cooling/heating source must be taken to get the ambient temperature within the DL405 operating specifications.
- 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.
- The DL405 system is designed to be powered by 110 VAC, 220 VAC, or 24 VDC normally available throughout an industrial environment. Isolation transformers and noise suppression devices are not normally necessary, but may be helpful in eliminating/reducing suspect power problems.



Enclosures

Your selection of a proper enclosure is important to ensure safe and proper operation of your DL405 system. Applications of DL405 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

Agency Approvals

Some applications require agency approvals. Typical agency approvals which your application may require are:

- UL (Underwriters' Laboratories, Inc.)
- CSA (Canadian Standards Association)
- CUL (Canadian Underwriters' Laboratories, Inc.)

To obtain the most current agency approval information, see the Agency Approval Checklist section on the specific component part number web page.

Environmental Specifications

The following table lists the environmental specifications that generally apply to the DL405 system (CPU, Bases, I/O Modules). The I/O module operation may fluctuate depending on the ambient temperature and your application. Please refer to the appropriate I/O modules specifications for the temperature derating curves applying to specific modules.

Specifications	Rating
Storage Temperature	–4° F to 158° F (–20° C to 70° C)
Ambient Operating Temperature	32° F to 140° F (0° C to 60° 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

*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

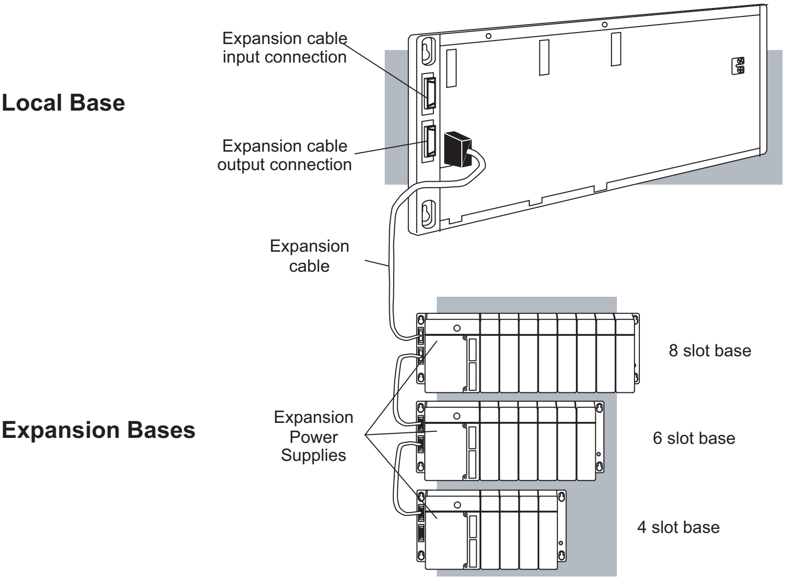
The power source must be capable of supplying voltage and current complying with the base power supply specifications.

Specifications	D4-454	D4-454DC-1
Input Voltage Nominal	120VAC	24VDC
Input Voltage Range	100–120 VAC; 196–240 VAC (+10% -15%)	20–29 VDC
Input Voltage Ripple	N/A	< 10%
Inrush Current, Maximum	20A	10A
Power Consumption, Maximum	50VA	38W
Voltage Withstand (dielectric)	1 min at 1500 VAC between primary, secondary, field ground and run relay	
Insulation Resistance	10MΩ at 500VDC	
Output Voltage auxiliary power supply	20–28 VDC (24V nominal), ripple more than 1V P-P	
Output Current auxiliary power supply	24VDC @ 400mA maximum	

Installing DL405 Bases

Three Sizes of Bases

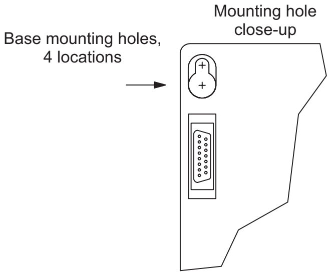
All I/O configurations of the DL405 will use a selection of either 4, 6, or 8 slot base(s). Local and expansion bases can be 4, 6, or 8 slot in size. Local and expansion bases differ only in how they are wired in a system.



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.

The CPU/Expansion Unit/Remote Slave must always be installed in the left-most slot in a base. This slot is marked on the base as P/S, CPU. The I/O modules can be installed in any remaining slots. It is not necessary for all slots to be filled for your system to work correctly. You may use filler modules to fill the empty slots in the base.

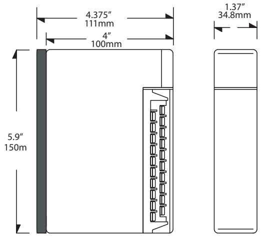
The base is secured to the equipment panel or machine using four M4 screws in the corner locations shown to the right. The mounting cut-outs allow removal of the base after installation, without completely removing the mounting screws. Full mounting template dimensions are given in the previous section on Mounting Guidelines.



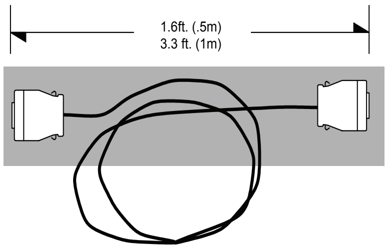
Component Dimensions

Before installing your PLC system you will need to know the dimensions for the components in your system. The diagram on this page provide the component dimensions and should be used to define your enclosure specifications. Remember to leave room for potential expansion. Please see the Product Weights Appendix for the weights for each component.

I/O modules

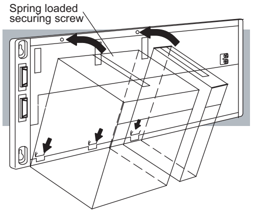


Base Expansion Cable



Installing Components in the Base

- The components have plastic tabs at the bottom and a screw that the top.
- With the device tilted slightly forward, hook the plastic tabs into the notch on the base.
- Then gently push the top of the component back toward the base until it is firmly installed into the base.
- Now tighten the screw at the top of the device to secure it to the base.



WARNING 1: To minimize the potential damage to the CPU/Module rear base connector(s), do not force the module into position. It should glide smoothly into place. If it does not glide smoothly into place, please check the following:
a) Visually inspect the base connector to make sure the mating holes are not blocked.
b) Visually inspect the pins on the rear connector making sure they are not bent or pushed in.
c) Make sure the plastic tabs on the bottom of the CPU/Module are level before tilting the module into position. Damage to the connectors may occur if the above information is not considered.
WARNING 2: To minimize the risk of electrical shock, personal injury or equipment damage, always disconnect the system power before installing or removing any system equipment.

CPU and Expansion Unit Wiring Guidelines

The main power terminal connections are under the front covers of the D4-454 CPUs and DL405 Expansion Units. The list below describes the function of each of the terminal screws. Most of the terminal screws are identical between the CPU and the Expansion unit. If the terminal screw only applies to one of the units it will be noted.

Run Relay - (CPU only) indicates to an external device when the CPU is in Run Mode by contact closure. Its normally-open contacts can also remove power from critical I/O points if CPU comes out of Run Mode.

24VDC Auxiliary Power - can be used to power field devices or I/O modules requiring external power. It supplies up to 400mA of current at 20–28 VDC, ripple less than 1 V P-P. (Not available on DC CPU model.)

Logic Ground - internal ground to the system which can be tied to field devices/ communication ports to unite ground signals.

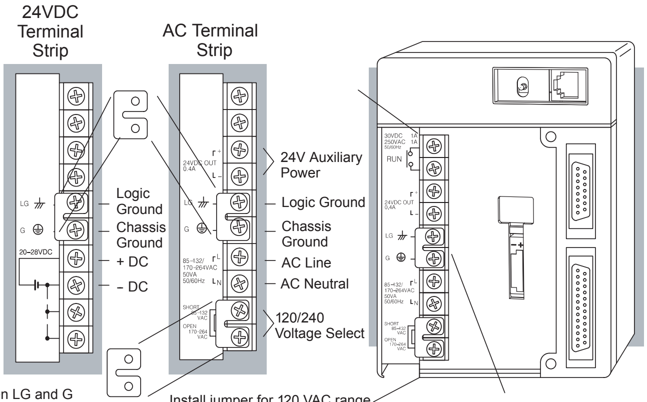
Chassis Ground - where earth ground is connected to the unit.

AC Power - where the line (hot) and the neutral (common) connections are made to the CPU/Expansion Unit. (This is also where the DC power source is connected for the 24VDC CPU. The positive connection is tied to line and the negative connection is tied to ground.

120/240 Voltage Select - a jumper across two of the terminals determines the voltage selection. Install the jumper to select 120VAC input power, and remove the jumper to select 240VAC power input (the jumper is not required for DC-powered CPUs or Expansion Units.)

WARNING: Damage will occur to the power supply if 240 VAC is connected to the terminal connections with the 120VAC jumper installed. Once the power wiring is connected, install the protective cover to avoid risk of accidental shock.

CPU Wiring



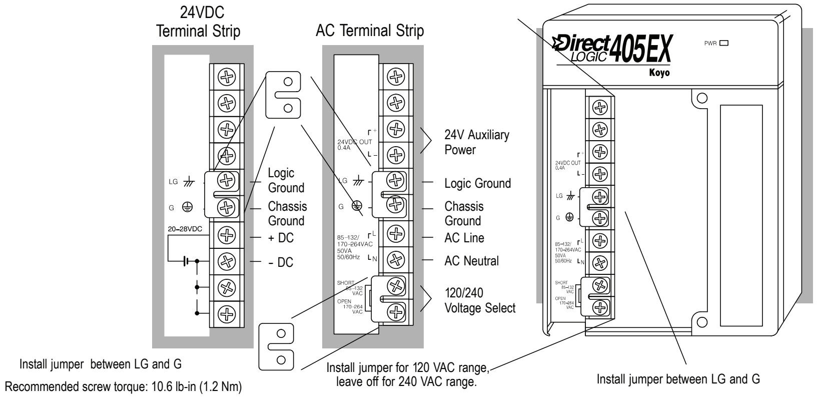
Install jumper between LG and G
Recommended **screw torque**: 10.6 lb–in (1.2Nm)

Install jumper for 120 VAC range, leave off for 240 VAC range.

Install jumper between LG and G

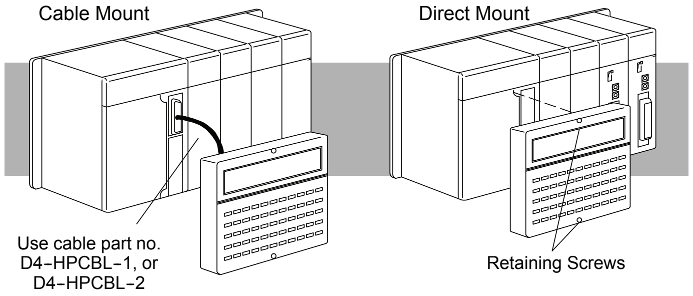
Expansion Unit Wiring

The following diagram details the appropriate connections for each terminal.



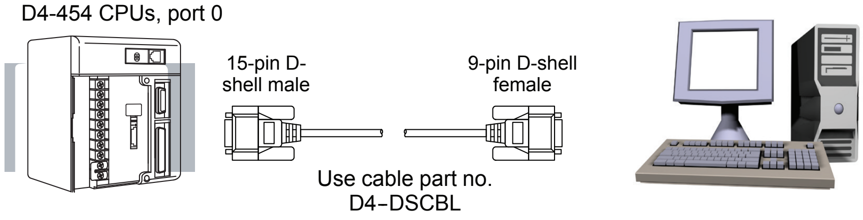
Connecting Programming Devices

You can mount the Handheld directly to Port 0 of the D4-454 CPU (15-Pin D-Shell connector), or you can use a 9ft (3m) or 4.6 ft (1.5 m) cable as shown below.

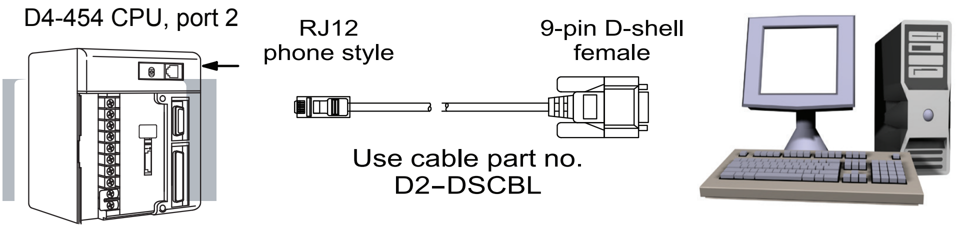


DirectSOFT Programming

The standard port for use in DirectSOFT programming is the 15-pin port 0 on both D4-454 CPUs. The cable shown below is approximately 12 feet (3.6m) long.

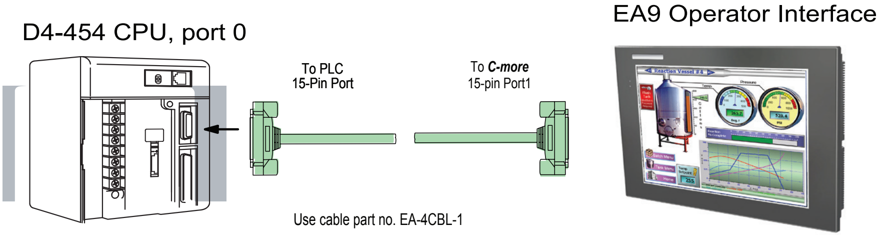


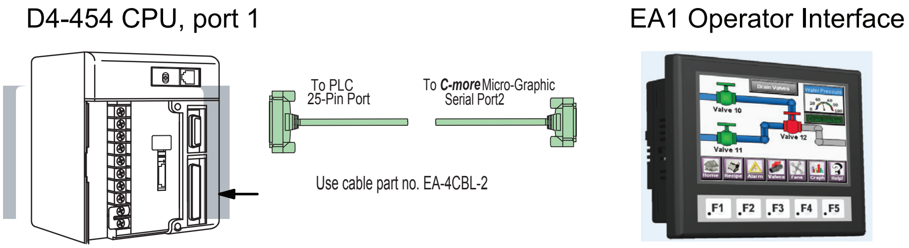
On the D4-454, you may use port 2 instead for DirectSOFT programming. The cable shown below is approximately 12 feet (3.66 m) long.

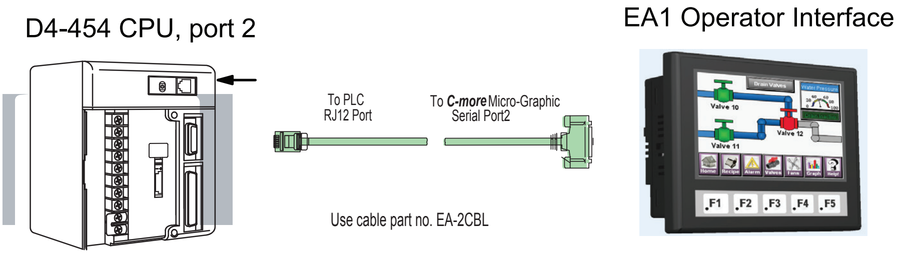


Connecting Operator Interface Devices

It is very common to connect an operator interface to DL405 PLCs. The C-more touch panels provide a graphical interface designed to display graphics and animation, and to exchange data to and from the PLC by means of the touch screen.





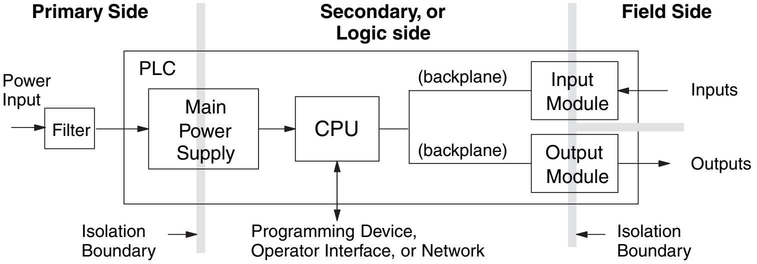


I/O Wiring Strategies

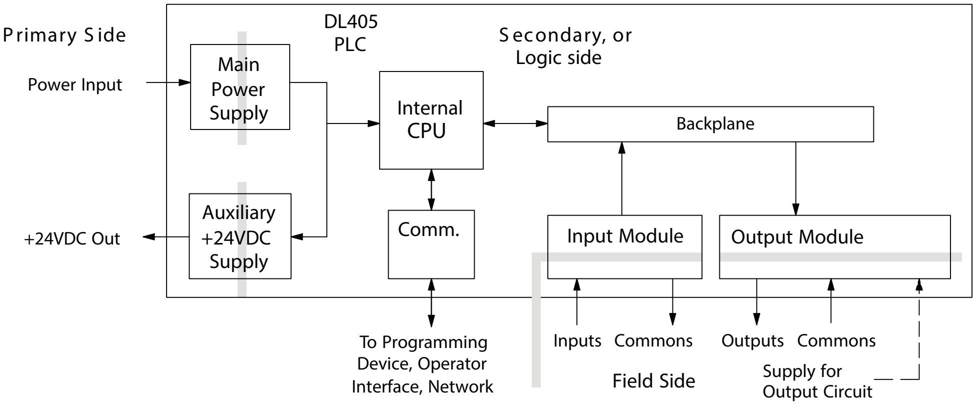
The DL405 PLC system is very flexible and will work in many different wiring configurations. By studying this section before actual installation, you can probably find the best wiring strategy for your application. This will help to lower system cost, wiring errors, and avoid safety problems.

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.



The next figure shows the physical layout of a DL405 PLC system, as viewed from the front. In addition to the basic circuits covered above, AC- powered CPUs include an auxiliary +24 VDC 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.

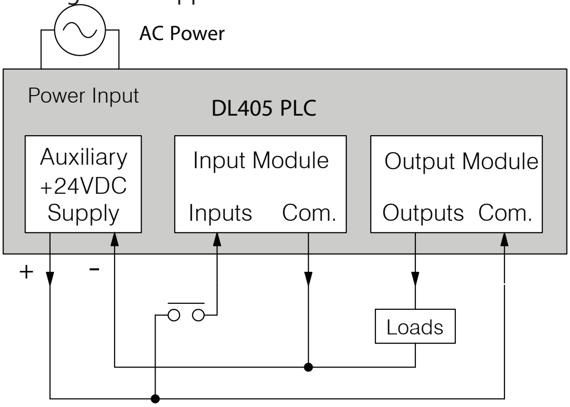


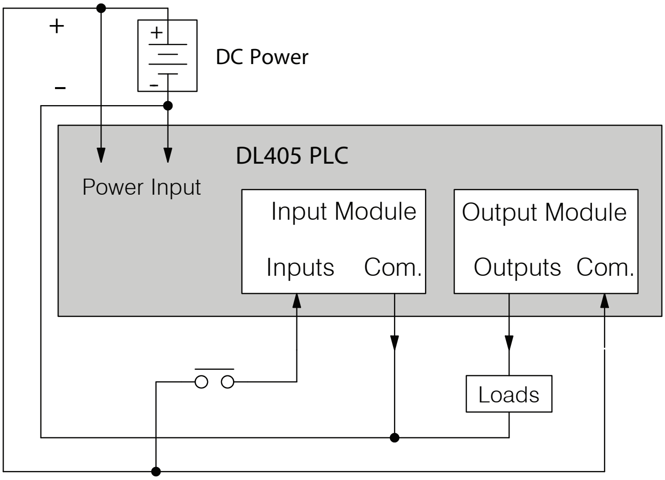
Powering I/O Circuits with the Auxiliary Supply

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 400mA. 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.

The D4-454 CPU features 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 (400mA limit).

The D4-454DC-1 is 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.

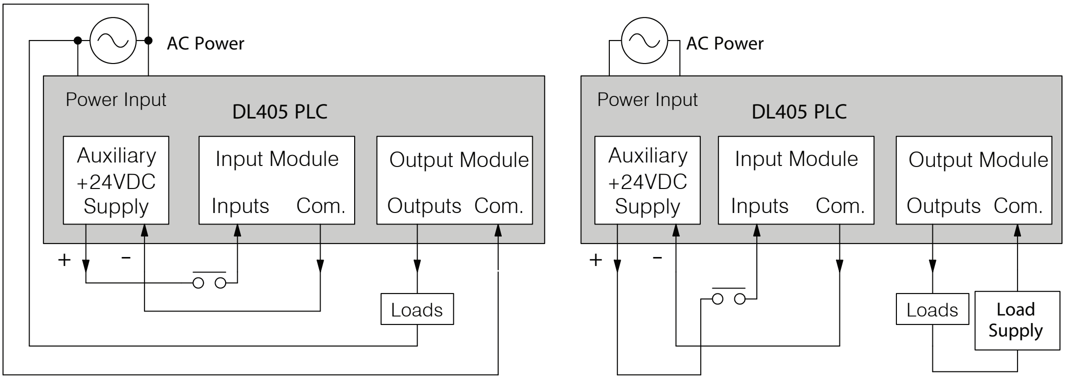




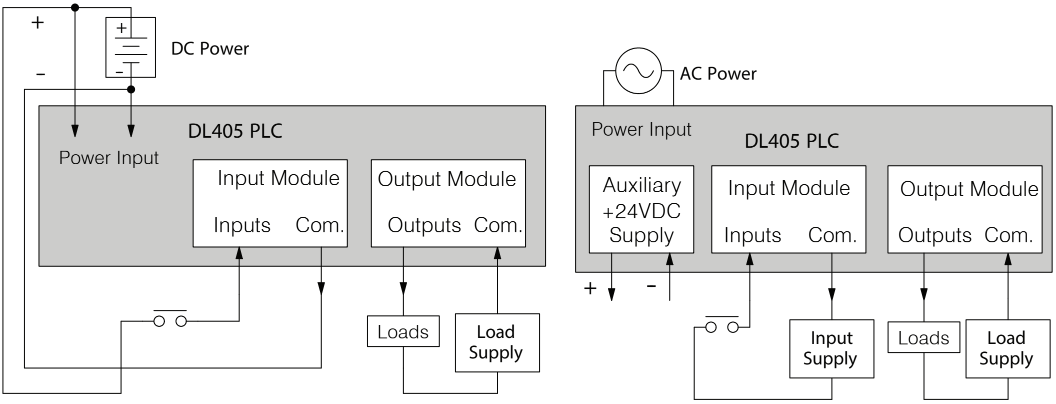
Powering I/O Circuits Using Separate Supplies

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

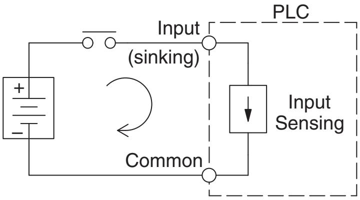
Before going further in the study of wiring strategies, you must have a solid understanding of "sinking" and "sourcing" concepts. Use of these terms occurs frequently in input or output circuit discussions. It is the goal of this section to make these concepts easy to understand, further ensuring your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

Sinking = provides a path to supply ground (–)

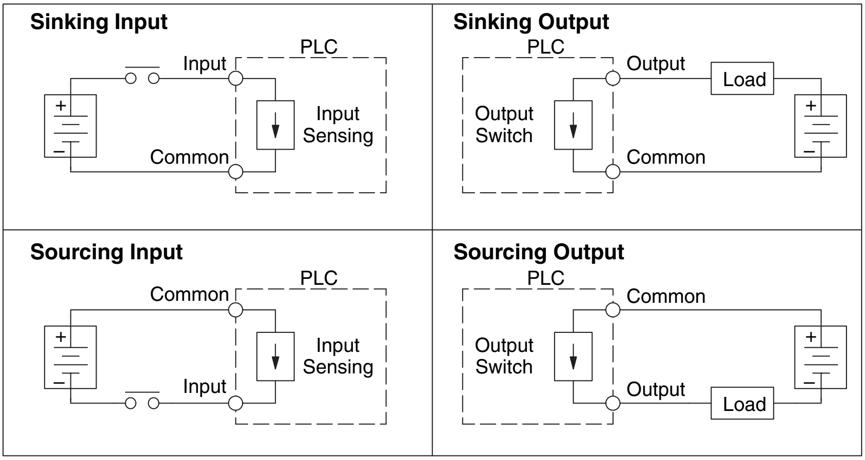
Sourcing = provides a path to supply source (+)

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.

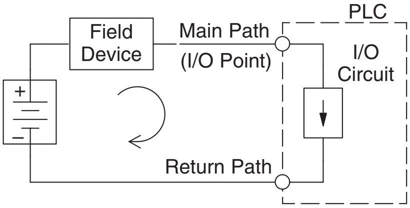


Apply the circuit principle above to the four possible combinations of input/output sinking/sourcing types as shown below. The I/O module specifications at the end of this chapter list the input or output type.

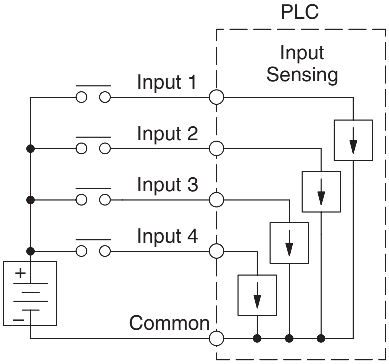


I/O "Common" Terminal Concepts

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.



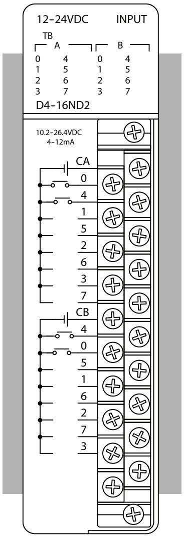
If there was unlimited space and budget for I/O terminals, every I/O point could have two dedicated terminals as the figure above shows. However, providing this level of flexibility is not practical or even necessary for most applications. So, most Input or Output points on PLCs are in groups which share 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.

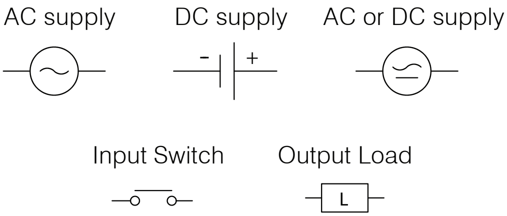


***NOTE:** In the circuit above, the current in the common path is 4 times any channel's input current when all inputs are energized. This is especially important in output circuits, where heavier gauge wire is sometimes necessary on commons*

Most DL405 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:



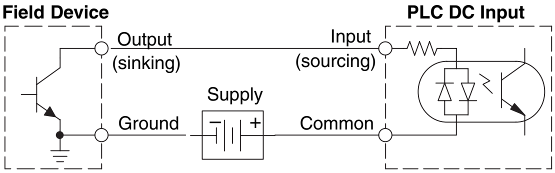


Connecting DC I/O to “Solid State” Field Devices

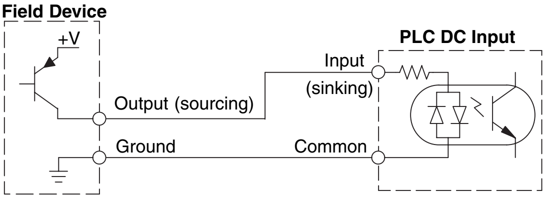
In the previous section on Sourcing and Sinking concepts, the DC I/O circuits were explained to sometimes only allow current to flow one way. This is also true for many of the field devices which have solid-state (transistor) interfaces. In other words, field devices can also be sourcing or sinking. 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 DL405 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 +24V auxiliary supply or another supply (+12VDC or +24VDC), as long as the input specifications are met.

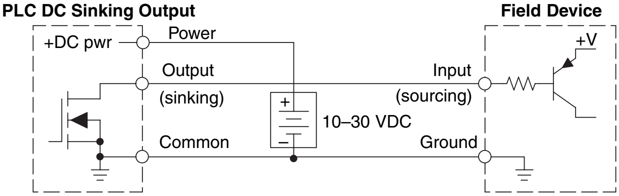


In the next circuit, a field device has an open-collector PNP transistor output. It sources current to the PLC input point, which sinks the current back to ground. Since the field device is sourcing current, no additional power supply is required.
Solid State Output Loads

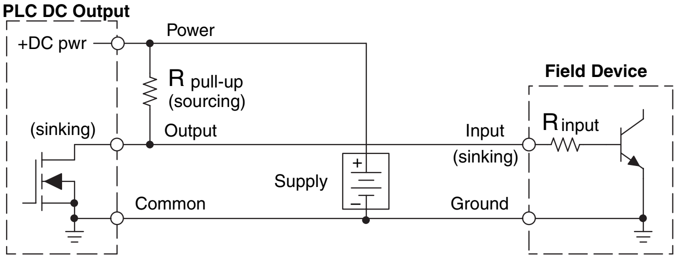


Sometimes an application requires connecting a PLC output point to a solid state input on a device. This type of connection is usually made to carry a low-level control signal, not to send DC power to an actuator.

Several of the DL405 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 Rpull-up is connected from the output to the DC output circuit power input.



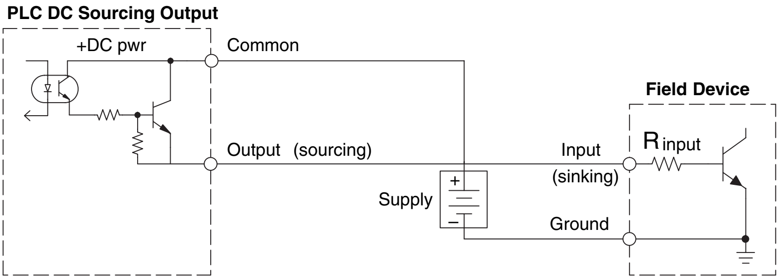
NOTE 1: *DO NOT attempt to drive a heavy load (> 25mA) with this pull-up method.*
NOTE 2: *Using the pull-up resistor to implement a sourcing output has the effect of inverting the output point logic. In other words, the field device input is energized when the PLC output is OFF, from a ladder logic 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 Rpull-up. In order to do so, you need to know the nominal input current to the field device (Iinput) when the input is energized. If this value is not known, it can be calculated as shown (a typical value is 15mA). Then use Iinput and the voltage of the external supply to compute Rpull-up. Then calculate the power Ppull-up (in watts), in order to size Rpull-up properly.

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

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

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 DL405 I/O family feature relay outputs: D4–08TR, F4-08TRS-1, F4-08TRS-2, D4-16TR. 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)

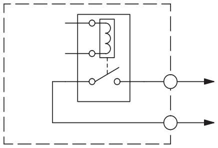
Some applications in which NOT to use relays:

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

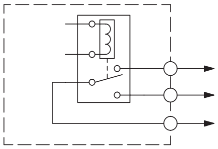
Relay outputs in the DL405 output modules are available in two contact arrangements, shown to the right. The Form A type, or SPST (single pole, single throw) type is normally open and is the simplest to use. The Form C type, or SPDT (single pole, double throw) type has a center contact which moves and a stationary contact on either side. This provides a normally closed contact and a normally open contact.

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 A contacts



Relay with Form C contacts



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

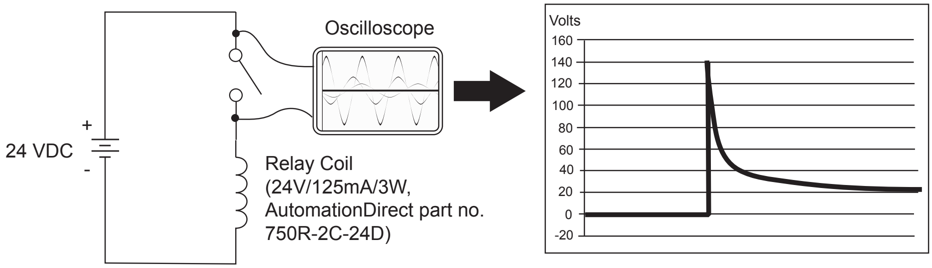
The following pages are intended to give a quick overview of the negative effects of transient voltages on a control system and provide some simple advice on how to effectively minimize them. The need for transient suppression is often not apparent to the newcomers in the automation world. Many mysterious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

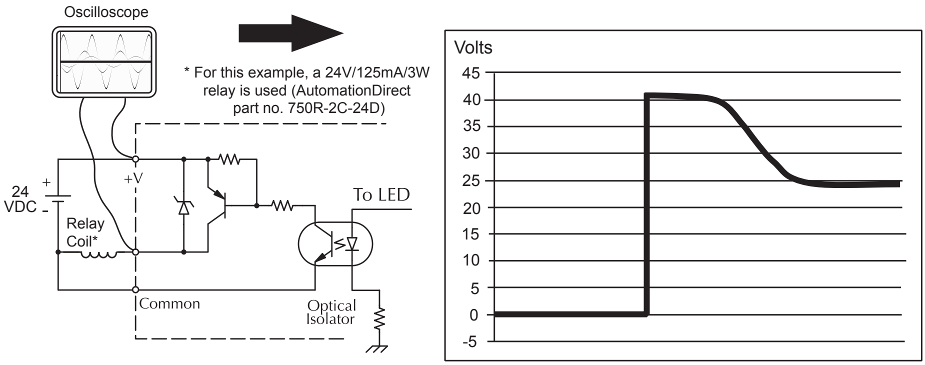
A very destructive side-effect of the arcing across relay contacts is the electromagnetic interference (EMI) it can cause. This occurs because the arcing causes a current surge, which releases RF energy. The entire length of wire between the relay contacts, the coil, and the power source carries the 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.

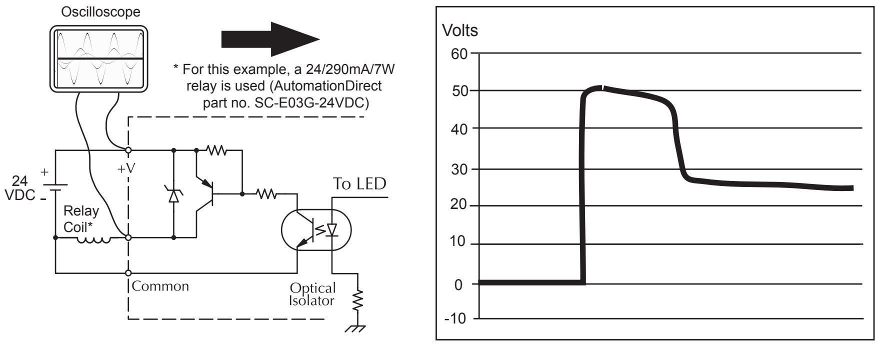
The next example uses the same 24V/125mA/3W relay used earlier. This example measures a PLC PNP transistor output, which incorporates an integrated Zener diode for transient suppression. Instead of the 140V peak in the first example, the transient voltage here is limited to about 40V by the Zener diode. 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


* For this example, a 24V/125mA/3W relay is used (AutomationDirect part no. 750R-2C-24D)

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


* For this example, a 24/290mA/7W relay is used (AutomationDirect part no. SC-E03G-24VDC)

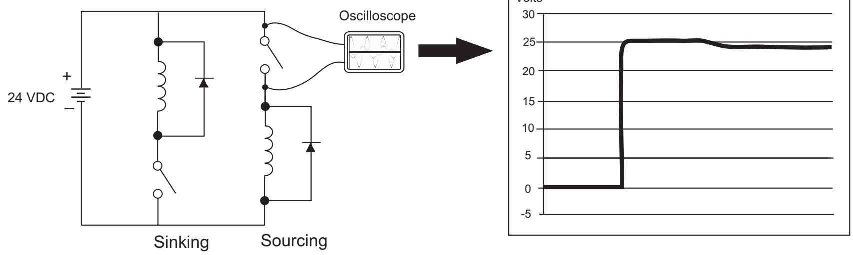
Additional transient suppression should be used in both these examples. If you are unable to measure the transients generated by the connected loads of your control system, using additional transient suppression on all inductive loads would be the safest practice.

Types of Additional Transient Protection

DC Coils:

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

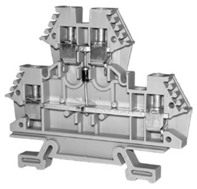
DC Flyback Circuit



Many AutomationDirect's socketed relays and motor starters have add-on fly back 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 fly back 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



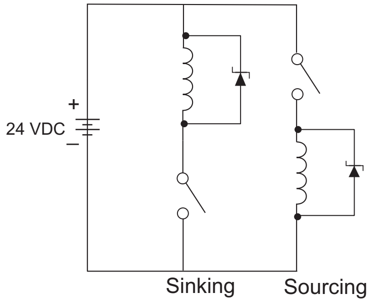
DN-D10DR-A
Diode Terminal Block

Two more common options for DC coils are Metal Oxide Varistors (MOV) or TVS diodes. These devices should be connected across the coil being driven (transient generator) 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.

ZL-TSD8-24
Transorb Module

DC MOV or TVS Diode Circuit



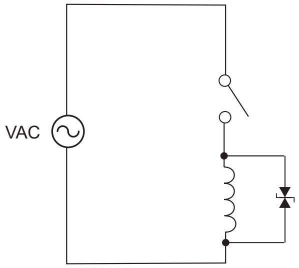
AC Coils:

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 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.

ZL-TSD8-120
Transorb Module

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 suppressors 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 Modules Wiring, and Specification

Slot Numbering

The DL405 bases come in 4, 6 or 8 slot sizes with the first slot on the far left dedicated for the CPU or one of the Base Controller Modules. The I/O and Specialty Modules can be placed in any of the remaining slots on the base. The CPU slot is NOT numbered. The first slot to the right of the CPU starts at zero. For example, the slot numbering on an 8 slot base (D4-08B-1) is slot 0 to 7.

Module Placement Restrictions

The following table lists the valid locations for all types of modules in a D4-454 system:

Module/Unit	Local CPU Base	Local Expansion Base	Remote I/O Base
CPUs	CPU Slot Only	–	–
Input Modules	✓	✓	✓
Output Modules	✓	✓	✓
Relay Output Modules	✓	✓	✓
Analog Modules	✓	✓	✓
Local Expansion			
D4-EX	–	✓	–
D4-EXDC	–	✓	–
Communications and Networking			
D4-DCM	✓ (*)	–	–
H4-ECOM100	✓	–	–
F4-MAS-MB	✓	–	–
Remote I/O			
H4-ERM100	✓	–	–
H4-EBC	–	–	✓
D4-RM	✓	–	–
D4-RS	–	–	✓
D4-RSDC	–	–	✓
Motion			
D4-HSC	✓	✓	–
H4-CTRIO	✓	✓	✓
Specialty Modules			
D4-16SIM	✓	✓	✓
F4-16PID	✓	–	–
F4-8MPI	✓	–	–
F4-4LTC	✓	–	–
F4-CP128	✓	–	–
F4-CP128-T	✓	–	–
D4-FILL	✓	✓	✓

(*) When used with H4-ERM100 Ethernet Remote I/O system

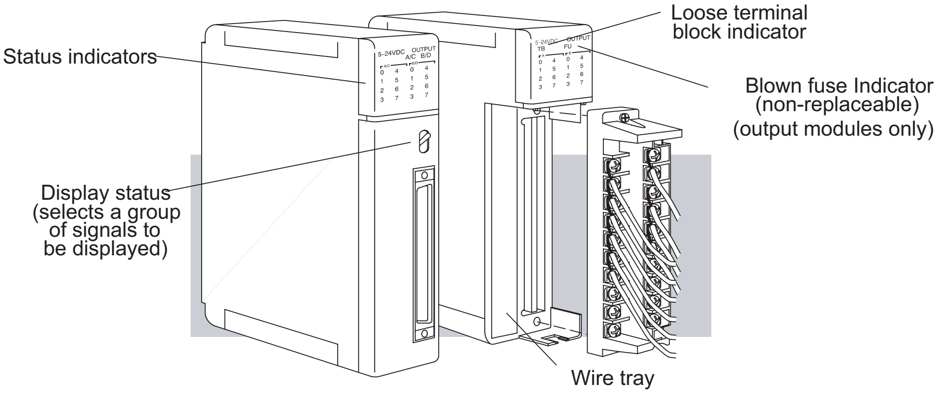
Module Placement

Before wiring the I/O modules in your system to field devices, it's very important to make sure each I/O module is in the right slot and base in the system. Costly wiring errors may be avoided by doing the following:

- Do the power budget calculations for each base to verify the base power supply can power all the modules in the base. Information on how to do this is in Chapter 4, System Design and Configuration.
- Some Specialty I/O modules may only be installed in particular slots (will not function properly, otherwise). Check the corresponding manual before installation and wiring.
- Whenever possible, keep modules with high voltage and current wiring away from sensitive analog modules.

I/O Module Status Indicators

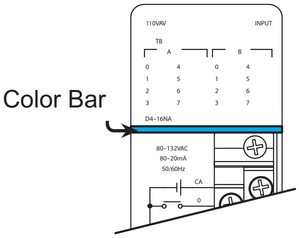
The discrete modules provide LED status indicators to show the status of the I/O points.


Loose terminal block indicator
Status indicators
Blown fuse Indicator (non-replaceable) (output modules only)
Display status (selects a group of signals to be displayed)
Wire tray
5–24VDC
OUTPUT
A/C B/D
TB
FU

Color Coding of I/O Modules

The DL405 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

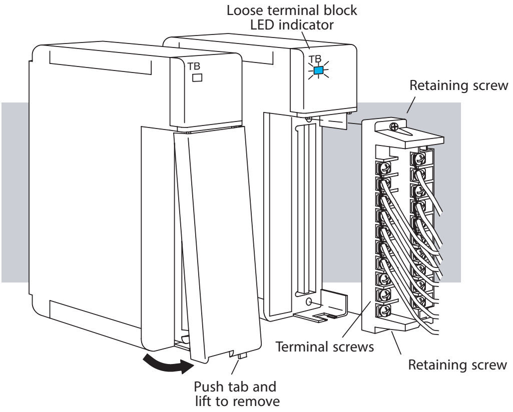


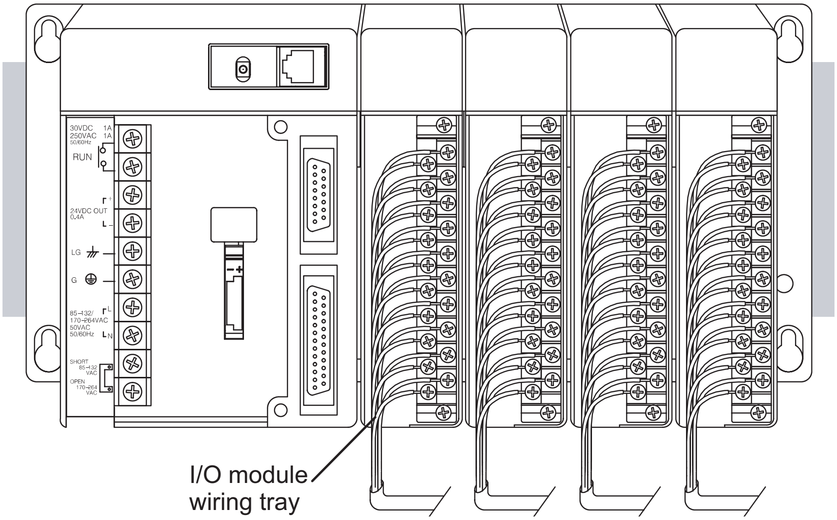
Wiring the Different Module Connectors

You must first remove the front cover of the module prior to wiring. To remove the cover depress the bottom tab of the cover and tilt the cover up to loosen from the module

All DL405 I/O module terminal blocks are removable for your convenience. To remove the terminal block loosen the retaining screws and lift the terminal block away from the module. When you return the terminal block to the module make sure the terminal block is tightly seated. Be sure to tighten the retaining screws. You should also verify the loose terminal block LED is off when system power is applied.

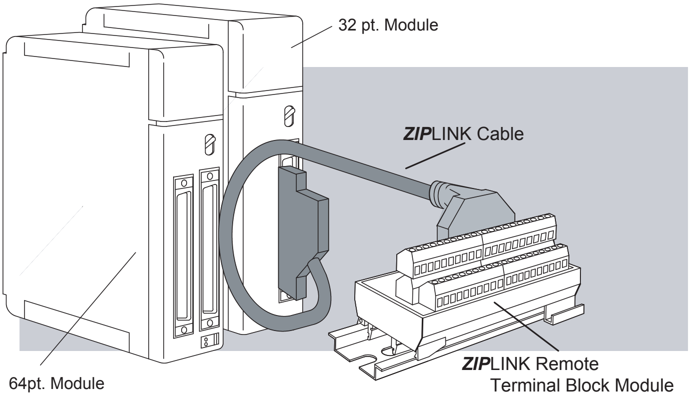
WARNING: For some modules, field device power may still be present on the terminal block even though the PLC system is turned off. To minimize the risk of electrical shock, check all field device power before you remove the connector.





Wiring 32 and 64 Point I/O Modules

The 32-point and 64-point I/O modules use a different style of connector due to the increased number of I/O points. A ZIPLink connection system is shown in the figure below. Another option is to use the D4-IOCBL-1, a 3 meter prewired solder connector and cable with pigtail.



The ZIPLink system offers “plug and play” capability, eliminating the need for traditional wiring. Simply plug one end of the ZIPLink cable into a 32 or 64 point I/O module and the other end into a ZIPLink Remote Terminal Block Module. Please review out online ZIPLink Selection Table to ensure proper module/cable combinations are chosen for your application.

I/O Wiring Checklist

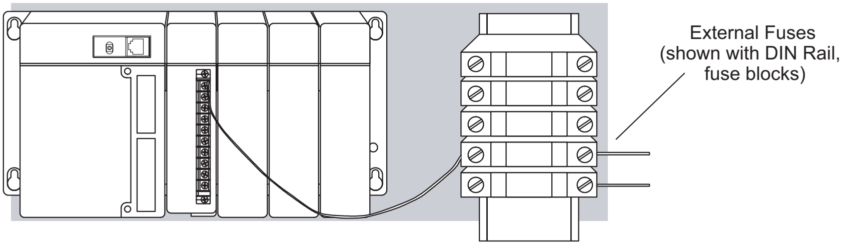
Use the following guidelines when wiring the I/O modules in your system.

Module type	Suggested AWG Range	Suggested Torque
CPU	12AWG	10.63 lb·inch (1.2 N·m)
8 Point	12AWG	7.97 lb·inch (0.9 N·m)
16 Point	14AWG	7.97 lb·inch (0.9 N·m)
32 Point 64 Point	*ZIP*Link Solution Required. Review out online selection table to ensure proper module/cable combinations are chosen. D4-IOCBL-1 (3m pigtail cable with D4-IO3264S)	

NOTE: *Wire Type TFFN or Type MTW is recommended. Other types of wire may be acceptable, but it really depends on the thickness of the wire insulation. If the insulation is too thick and you use all the I/O points, then the plastic terminal cover may not close properly.*

- There is a limit to the size of wire the modules can accept. The table above lists the suggested AWG for each module type. When making terminal connections, follow the suggested torque values.
- Always use a continuous length of wire, do not combine wires to attain a needed length.
- Use the shortest possible wire length.
- Use wire trays for routing where possible.
- Avoid running wires near high energy wiring. Also, avoid running input wiring close to output wiring where possible.
- To minimize voltage drops when wires must run a long distance , consider using multiple wires for the return line.
- Avoid running DC wiring in close proximity to AC wiring where possible.
- Avoid creating sharp bends in the wires.

NOTE: *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 DIN rail mounted fuse blocks.*



DL405 Input Module Chart

The following table lists the available DL405 input modules. Specifications begin on the following page.

DL405 Input Module Types	Number of Input Points	DC Current Sink Input	DC Current Source Input	AC Input
D4-16ND2	16	-	✓	-
D4-16ND2F	16	-	✓	-
D4-32ND3-1	32	✓	✓	-
D4-32ND3-2	32	✓	✓	-
D4-64ND2	64	-	✓	-
D4-08NA	8	-	-	✓
D4-16NA	16	-	-	✓
D4-16NE3	16	✓	✓	✓
F4-08NE3S	8	✓	✓	✓
D4-08ND3S	8	✓	✓	-

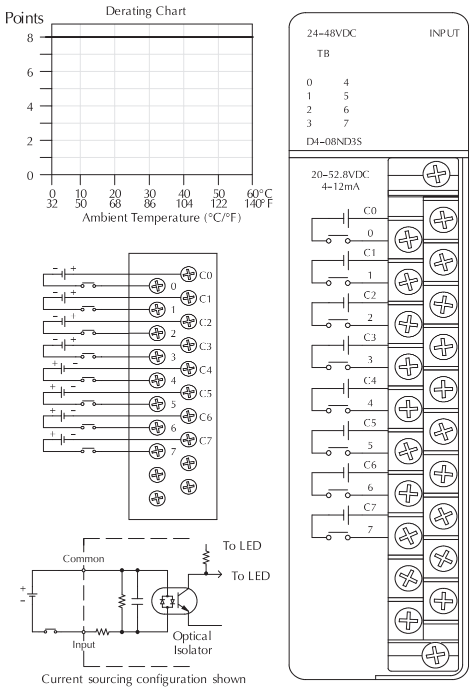
DL405 Output Module Chart

The following table lists the available DL405 output modules. Specifications begin on the following page.

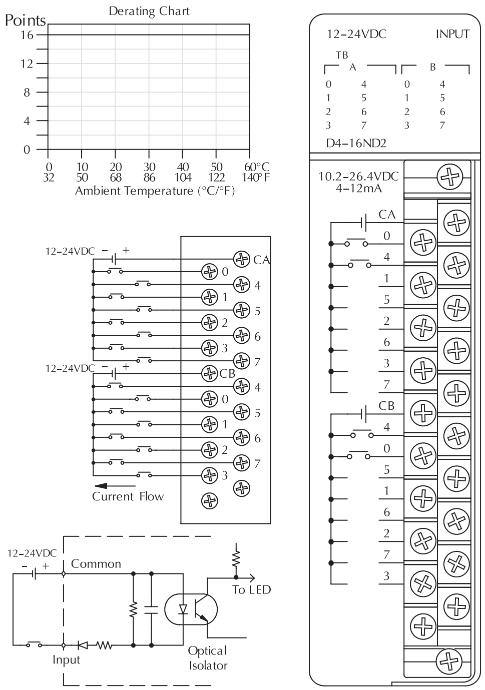
DL405 Output Module Types	Number of Output Points	DC Current Sink Output	DC Current Source Output	AC Output
D4-08TD1	8	✓	-	-
F4-08TD1S	8	✓	-	-
D4-16TD1	16	✓	-	-
D4-16TD2	16	-	✓	-
D4-32TD1	32	✓	-	-
D4-32TD1-1	32	✓	-	-
D4-32TD2	32	-	✓	-
D4-64TD1	64	✓	-	-
D4-08TA	8	-	-	✓
D4-16TA	16	-	-	✓
D4-08TR	8	✓	✓	✓
F4-08TRS-1	8	✓	✓	✓
F4-08TRS-2	8	✓	✓	✓
D4-16TR	16	✓	✓	✓

D4-08ND3S DC Input	
Inputs Per Module	8 (sink/source)
Commons Per Module	8 (isolated)
Input Voltage Range	20–52.8 VDC
Peak Voltage	52.8 VDC
On Voltage Level	> 18V
Off Voltage Level	< 7V
Input Impedance	4.8 kΩ
Input Current	5mA @ 24VDC 10mA @ 48VDC
Minimum On Current	3.5 mA
Maximum Off Current	1.5 mA
Base Power Required 5V	100mA max
OFF to ON Response	3–10 ms
ON to OFF Response	3–12 ms
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side

D4-16ND2 DC Input	
Inputs Per Module	16 (current sourcing)
Commons Per Module	2 (isolated)
Input Voltage Range	10.2–26.4 VDC
Peak Voltage	26.4 VDC
On Voltage Level	> 9.5 V
Off Voltage Level	< 4.0 V
Input Impedance	3.2 kΩ @ 12VDC 2.9 kΩ @ 24VDC
Input Current	3.8 mA @ 12VDC 8.3 mA @24VDC
Minimum On Current	3.5 mA
Maximum Off Current	1.5 mA
Base Power Required 5V	150mA max
OFF to ON Response	1–7ms (2.3 typical)
ON to OFF Response	2–12ms (4.6 typical)
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side

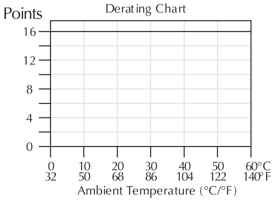


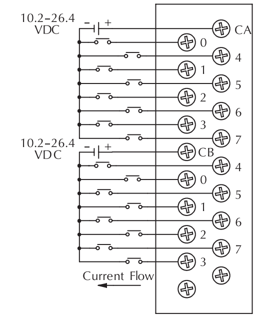
Current sourcing configuration shown

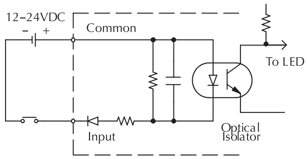


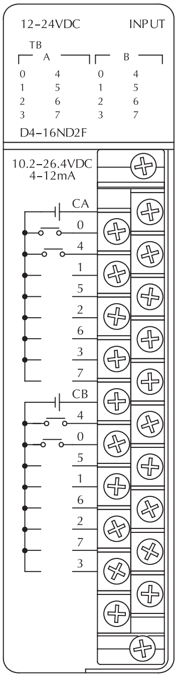
D4-16ND2F DC Input	
Inputs Per Module	16 (current sourcing)
Commons Per Module	2 (isolated)
Input Voltage Range	10.2–26.4 VDC
Peak Voltage	26.4 VDC
On Voltage Level	> 9.5V
Off Voltage Level	< 4.0V
Input Impedance	3.2 kΩ @ 12VDC 2.9 kΩ @ 24VDC
Input Current	3.8 mA @ 12VDC 8.3 mA @ 24VDC
Minimum On Current	3.5 mA
Maximum Off Current	1.5 mA
Base Power Required 5V	150mA max
OFF to ON Response	1ms
ON to OFF Response	1ms
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side

D4-16 SIM Input Simulator	
Inputs per Module	8 or 16 selectable by internal switch
Base Power Required 5V	150mA max
Terminal Type	None
Status Indicators	Logic Side



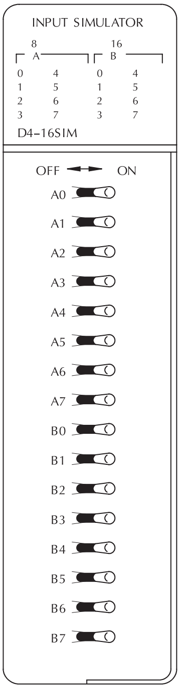






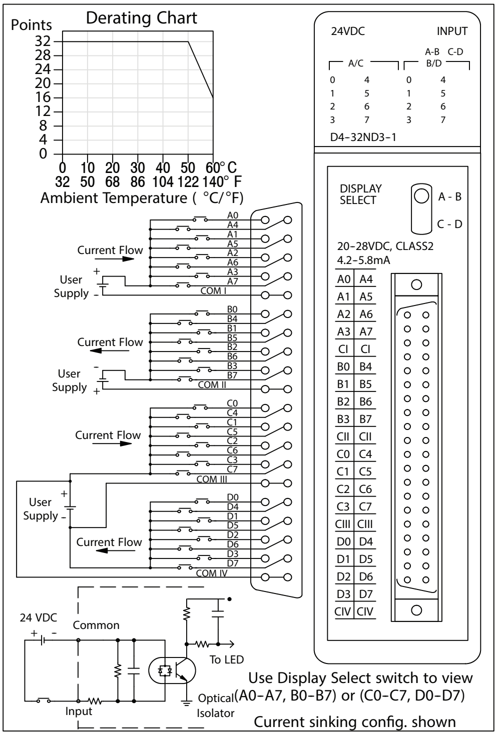
8 or 16 input point selection is located on the back of the module.

Switch position is indicated by the LEDs above the input switches.



D4-32ND3-1 DC Input	
Inputs Per Module	32 (sink/source)
Commons Per Module	4 (isolated)
Input Voltage Range	20–28 VDC
Peak Voltage	30VDC
On Voltage Level	> 19V
Off Voltage Level	< 10V
Input Impedance	4.8 kΩ
Input Current	5mA @ 24VDC
Minimum On Current	3.5 mA
Maximum Off Current	1.6 mA
Base Power Required 5V	150mA max
OFF to ON Response	2–10ms
ON to OFF Response	2–10m
Terminal Type	Connector sold separately *
Status Indicators	Logic Side

* *ZIP*Link prewired cable solution is recommended. See the online *ZIP*Link selector table for more information.



Use Display Select switch to view (A0–A7, B0–B7) or (C0–C7, D0–D7)

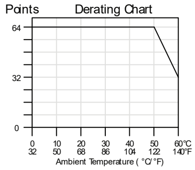
Current sinking config. shown

D4-64ND2 DC Input	
Inputs Per Module	64 (current sourcing)
Commons Per Module	8 (isolated)
Input Voltage Range	20–28 VDC
Peak Voltage	30VDC
On Voltage Level	> 20V
Off Voltage Level	< 13V
Input Impedance	4.8 kΩ
Input Current	5mA @ 24VDC
Minimum On Current	3.6 mA
Maximum Off Current	2.6 mA
Base Power Required 5V	300mA max

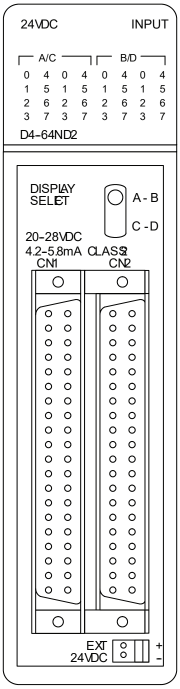
If you are using 64-pt. modules, you cannot install any specialty modules in slots 5, 6, or 7 of the local CPU base.

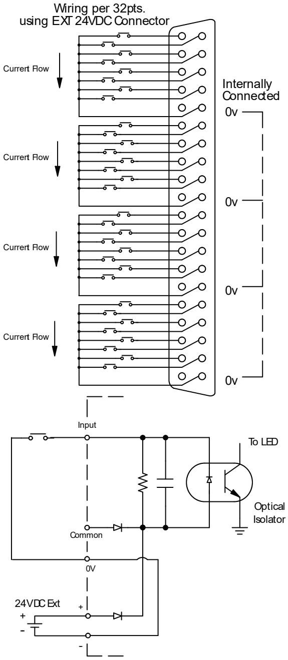
External Power (Optional)	24VDC ± 10%, 320mA max
OFF to ON Response	2 .5 ms (typical)
ON to OFF Response	5ms (typical)
Terminal Type	Connector sold separately *
Status Indicators	Logic Side

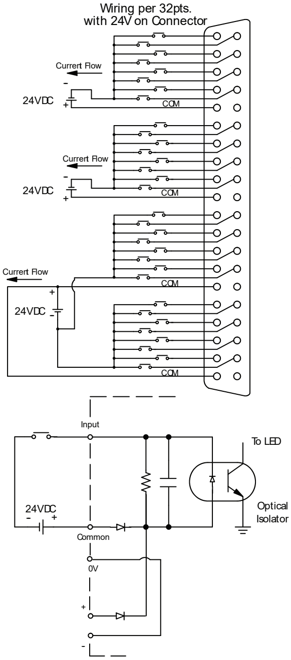
*ZIPLink prewired cable solution is recommended. See the online ZIPLink selector table for more information.

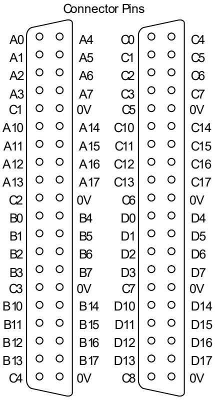


Since there are only 32 LEDs on the module, you can only display the status for 32 points at one time. In the A - B position the status of the first group of 32 input points (A0–A17, B0–B17) are displayed (connector 1). In the C - D position the status of the second group of 32 input points (C0–C17, D0–D17) are displayed (connector 2).





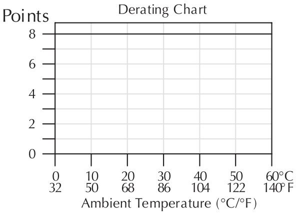


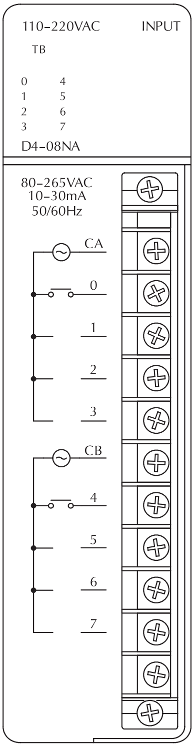


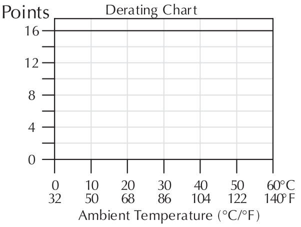
* Module location - this module placement is restricted to the local base on DL430/DL440 systems. It may also be placed in expansion bases in D4-450 and D4-454 system that are using the new (-1) bases.

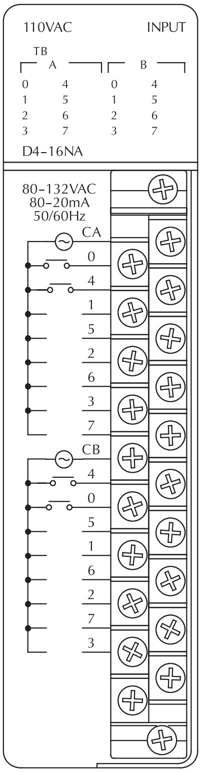
D4-08NA AC Input	
Inputs Per Module	8
Commons Per Module	2 (isolated)
Input Voltage Range	80–265 VAC
Peak Voltage	265VAC
AC Frequency	47–63 Hz
On Voltage Level	> 70V
Off Voltage Level	< 30V
Input Impedance	12kΩ
Input Current	8.5 mA @ 100VAC 20mA @ 230VAC
Minimum On Current	5mA
Maximum Off Current	2mA
Base Power Required 5V	100mA max
OFF to ON Response	5–30 ms
ON to OFF Response	10–50 ms
Terminal Type (Included)	Removable (D4-8IOCON)
Status Indicators	265VAC Logic Side

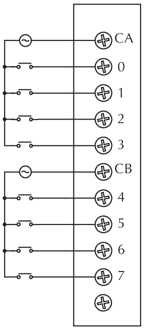
D4-16NA AC Input	
Inputs Per Module	16
Commons Per Module	2 (isolated)
Input Voltage Range	80–132 VAC
Peak Voltage	132VAC
AC Frequency	47–63 Hz
On Voltage Level	> 70V
Off Voltage Level	< 20V
Input Impedance	8kΩ
Input Current	14.5 mA @ 120VAC
Minimum On Current	7mA
Maximum Off Current	2mA
Base Power Required 5V	150mA max
OFF to ON Response	5–30 ms
ON to OFF Response	10–50 ms
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side

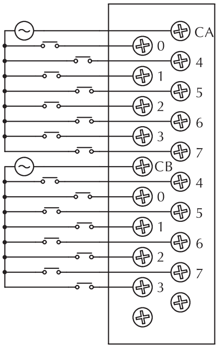


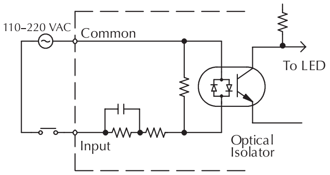


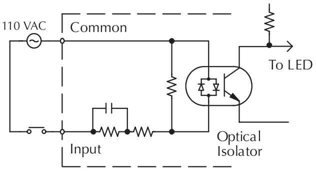






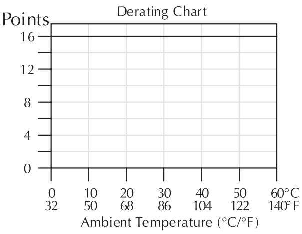


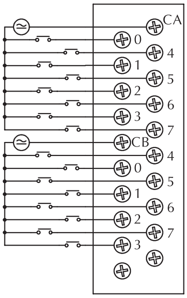


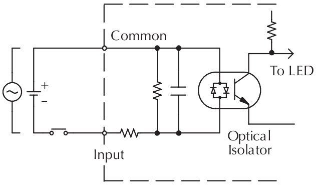


D4-16NE3 AC/DC Input	
Inputs Per Module	16 (sink/source)
Commons Per Module	2 (isolated)
Input Voltage Range	10.2–26.4 VAC/VDC
Peak Voltage	37.5 VAC/VDC
AC Frequency	47–63 Hz
On Voltage Level	> 9.5V
Off Voltage Level	< 3.0V
Input Impedance	3.2 kΩ @ 12V 2.9 kΩ @ 24V
Input Current	3.8 mA @ 12V 8.3 mA @ 24V
Minimum On Current	4mA
Maximum Off Current	1.5 mA
Base Power Required 5V	150mA max
OFF to ON Response	5–40 ms
ON to OFF Response	10–50 ms
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side

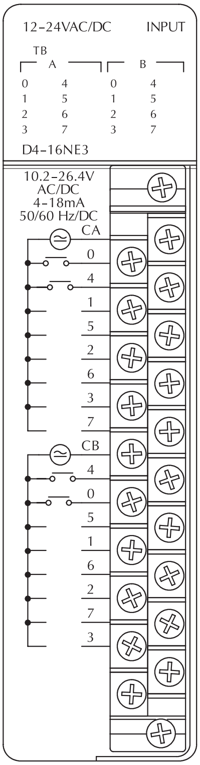
F4-08NE3S AC/DC Input	
Inputs Per Module	8 (sink/source)
Commons Per Module	8 (isolated)
Input Voltage Range	90–150 VAC/VDC
Peak Voltage	350 peak < 1ms
AC Frequency	47–63 Hz
On Voltage Level	> 90 VDC/75 VAC
Off Voltage Level	< 60 VDC/45 VAC
Input Impedance	22kΩ
Input Current	5.5 mA @ 120V
Minimum On Current	4mA
Maximum Off Current	2mA
Base Power Required 5V	90mA max
OFF to ON Response	8ms
ON to OFF Response	15ms
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side

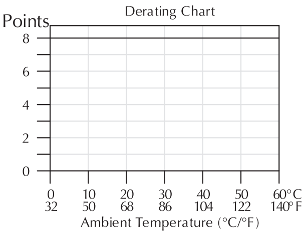


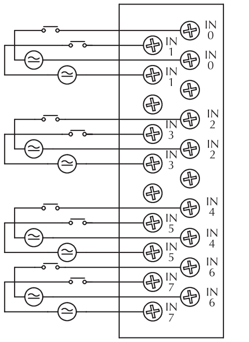


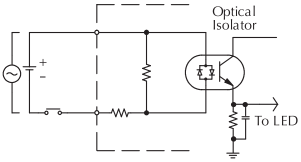


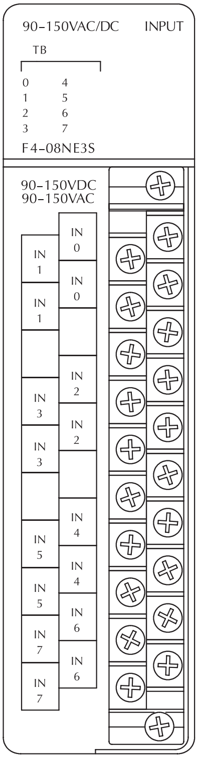
Current sourcing configuration shown



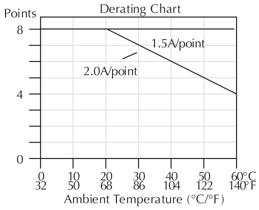


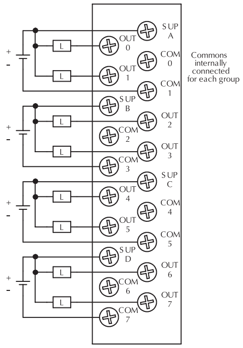


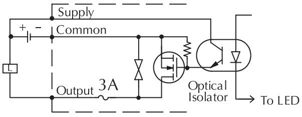


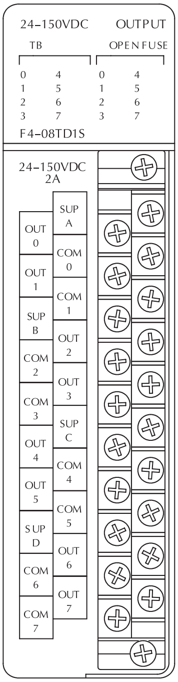


F4-08TD1S DC Output	
Outputs Per Module	8 (current sinking)
Commons Per Module	4 (isolated, 8 terminals)
Operating Voltage	24–150 VDC
Output Type	MOS FET
Peak Voltage	47–63 Hz
On Voltage Drop	0.5 VDC @ 2A
Max Current (resistive)	2A/point 4A/common
Max Leakage Current	5μA
Max Inrush Current	30A for 1ms 19A for 10ms
Minimum Load	N/A
Base Power Required 5V	295mA max
External DC Required	None
OFF to ON Response	25μs
ON to OFF Response	25μs
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side
Fuses	1 (3A) per output (see diagram) Non-replaceable



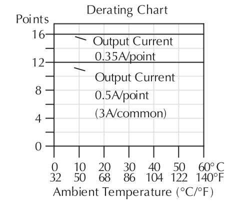


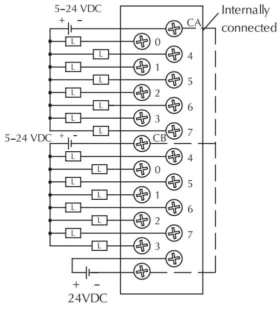


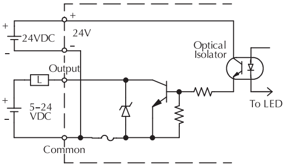


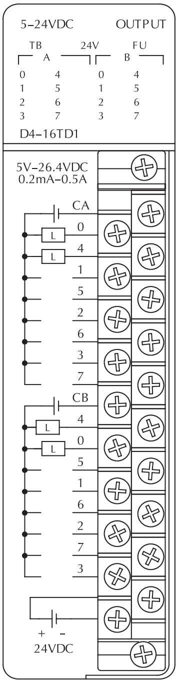
D4-16TD1 DC Output	
Outputs Per Module	16 (current sinking)
Commons Per Module	2 internally connected
Operating Voltage	4.5–26.4 VDC
Output Type	NPN Open Collector
Peak Voltage	40VDC
On Voltage Drop	0.5 VDC @ 0.5 A 0.2 VDC @ 0.1 A
Max Current (resistive)	0.5 A/point 3A/common
Max Leakage Current	0.1 mA @40VDC
Max Inrush Current	2A for 10ms 1A for 100ms
Minimum Load	0.2 mA
Base Power Required 5V	200mA max
External DC Required	24VDC ± 10% @125mA
OFF to ON Response	0.5 ms
ON to OFF Response	0.5 ms
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side
Fuses	1 (5A) per common Non-replaceable

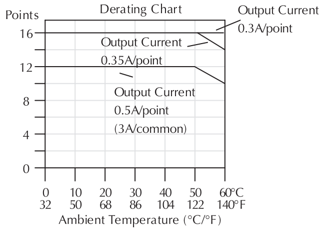
D4-16TD2 DC Output	
Outputs Per Module	16 (current sourcing)
Commons Per Module	2 (isolated)
Operating Voltage	10.2–26.4 VDC
Output Type	NPN Emitter Follower
Peak Voltage	40VDC
On Voltage Drop	1.5 VDC @ 0.5 A
Max Current (resistive)	0.5 A/point 3A/common @ 50°C 2.5 A/common @ 60°C
Max Leakage Current	0.1 mA @40VDC
Max Inrush Current	2A for 10ms 1A for 100ms
Minimum Load	0.2 mA
Base Power Required 5V	200mA max
External DC Required	None
OFF to ON Response	1ms
ON to OFF Response	1ms
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side
Fuses	1 (5A) per common Non-replaceable

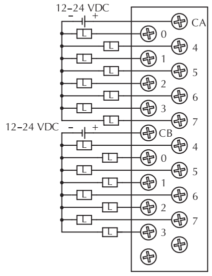


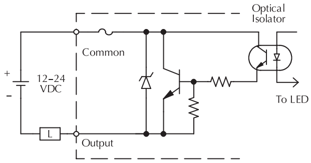


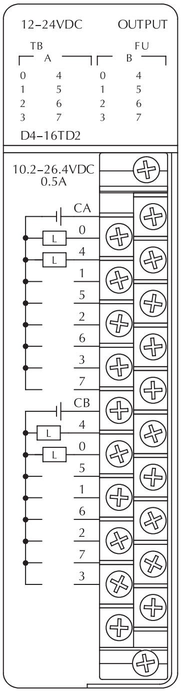










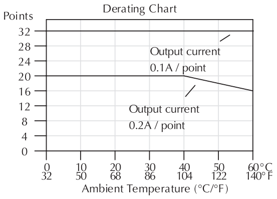


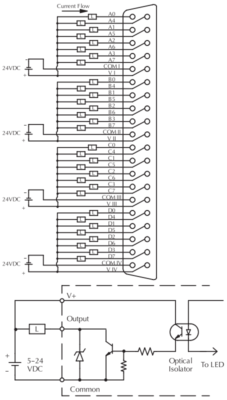
D4-32TD1 DC Output	
Outputs Per Module	32 (current sinking)
Commons Per Module	4 (isolated)
Operating Voltage	4.75–26.4 VDC
Output Type	NPN Open Collector
Peak Voltage	36VDC
On Voltage Drop	0.6 VDC @ 0.2 A
Max Current (resistive)	0.2 A/point 1.6 A/common
Max Leakage Current	0.1m A @ 36VDC
Max Inrush Current	1A for 10ms 0.5 A for 100ms
Minimum Load	0.1 mA
Base Power Required 5V	250mA max
External DC Required	24VDC ± 10%, 140mA max
OFF to ON Response	0.1 ms
ON to OFF Response	0.1 ms
Terminal Type	Connectors sold separately*
Status Indicators	Logic Side
Fuses	None

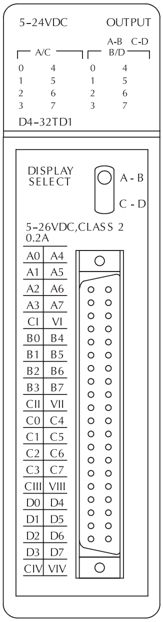
* ZIPLink prewired cable solution is recommended. See the online ZIPLink selector guide for more information.

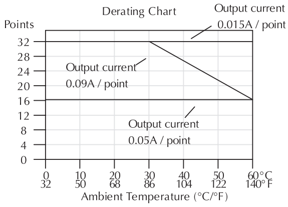
D4-32TD1-1 DC Output	
Outputs Per Module	32 (current sinking)
Commons Per Module	4 (isolated)
Operating Voltage	5–15 VDC
Output Type	NPN Open Collector (with pull-up)
Peak Voltage	16.5 VDC
On Voltage Drop	0.4 VDC @ 0.1 A
Max Current (resistive)	0.09 A/point 0.72 A/common 2.88 A/module
Max Leakage Current	0.1 mA @ 16.5 VDC
Max Inrush Current	0.5 A for 10ms 0.2 A for 100ms
Minimum Load	0.15 mA
Base Power Required 5V	250mA max
External DC Required	5–15 VDC ± 10%, 150mA max
OFF to ON Response	0.1 ms
ON to OFF Response	0.1 ms
Terminal Type (Included)	Connectors sold separately*
Status Indicators	Logic Side
Fuses	None

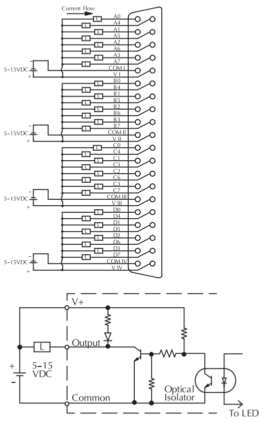
* ZIPLink prewired cable solution is recommended. See the online ZIPLink selector guide for more information.

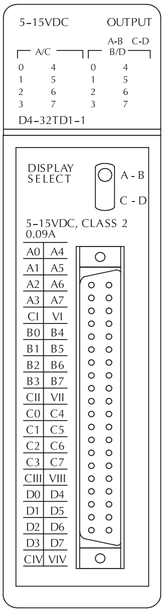






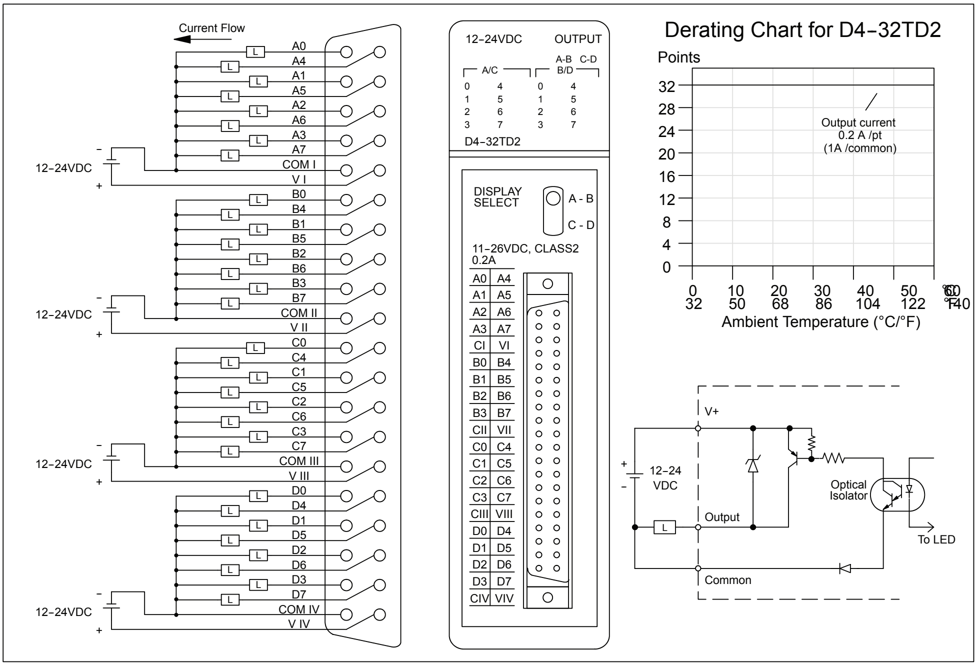






D4-32TD2 DC Output	
Outputs Per Module	32 (current sourcing)
Commons Per Module	4 (isolated)
Operating Voltage	10.8–26.4 VDC
Output Type	PNP Open Collector
Peak Voltage	30VDC
On Voltage Drop	0.6 VDC @ 0.2 A
Max Current (resistive)	0.2A/point 1A/common 4/module
Max Leakage Current	0.01 mA @ 26.4 VDC
Max Inrush Current	500mA for 10ms
Minimum Load	0.2 mA
Base Power Required 5V	350mA max
External DC Required	10.8–26.4 VDC 1A/common including load
OFF to ON Response	0.2 ms
ON to OFF Response	0.2 ms
Terminal Type	Connectors sold separately*
Status Indicators	Logic Side
Fuses	None

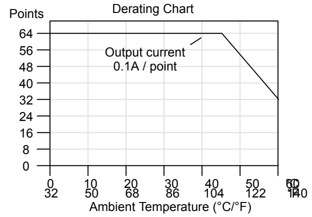
* ZIPLink prewired cable solution is recommended. See the online ZIPLink selector guide for more information.



D4-64TD1 DC Output	
Inputs Per Module	64 (current sinking)
Commons Per Module	8 (non-isolated)
Operating Voltage	4.75–28 VDC
Output Type	NPN Open Collector
Peak Voltage	36VDC
On Voltage Drop	0.6 VDC @0.1A
Max Current (Resistive)	0.1A/point 1A/common 7A per module total
Max Leakage Current	0.01mA @ 36VDC
Max Inrush Current	1A for 1ms, 700mA for 100ms
Minimum Load	0.1 mA

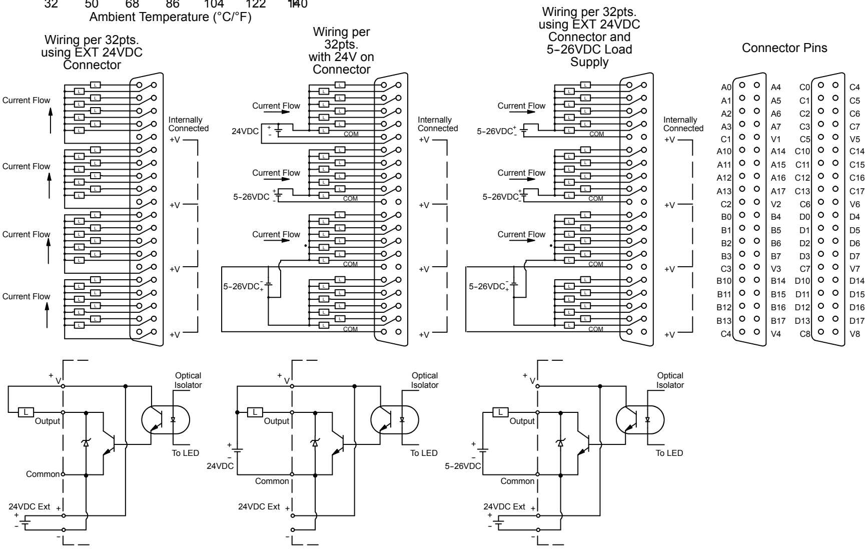
Base Power Required 5V	800mA max
External DC Required	24VDC ± 10%, (850mA per common) 7A total max
OFF to ON Response	0.1 ms
ON to OFF Response	0.2 ms
Terminal Type	Connector sold separately *
Status Indicators	Logic Side
Fuses	None

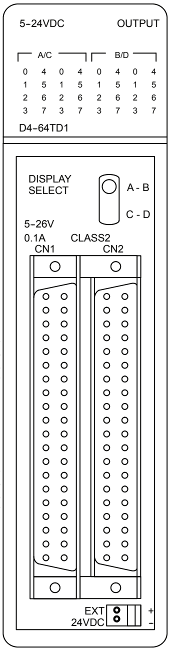
*ZIPLink prewired cable solution is recommended. See the online ZIPLink selector table for more information.



Only 32 status points can be displayed at one time on the front of the module. In the A - B position the status of the first group of 32 output points (A0–A17, B0–B17) are displayed (connector 1). In the C - D position the status of the second group of 32 output points (C0–C17, D0–D17) are displayed (connector 2).

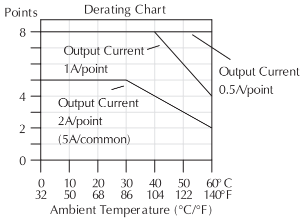
* Module location – this module placement is restricted to the local base on DL430/DL440 systems. It may also be placed in expansion bases in D4-450 and D4-454 systems that are using the new (–1) bases.

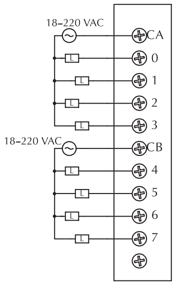

Wiring per 32pts. using EXT 24VDC Connector
Wiring per 32pts. with 24V on Connector
Wiring per 32pts. using EXT 24VDC Connector and 5–26VDC Load Supply
Connector Pins
Current Flow
Internally Connected
+V
24VDC
5–26VDC
COM
L
A0 A1 A2 A3 C1 A10 A11 A12 A13 C2 B0 B1 B2 B3 C3 B10 B11 B12 B13 C4
A4 A5 A6 A7 V1 A14 A15 A16 A17 V2 B4 B5 B6 B7 V3 B14 B15 B16 B17 V4
C0 C1 C2 C3 C5 C10 C11 C12 C13 C6 D0 D1 D2 D3 C7 D10 D11 D12 D13 C8
C4 C5 C6 C7 V5 C14 C15 C16 C17 V6 D4 D5 D6 D7 V7 D14 D15 D16 D17 V8
Optical Isolator
Output
Common
To LED
24VDC Ext
V

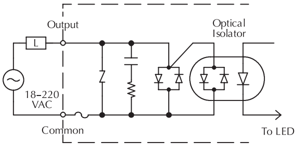


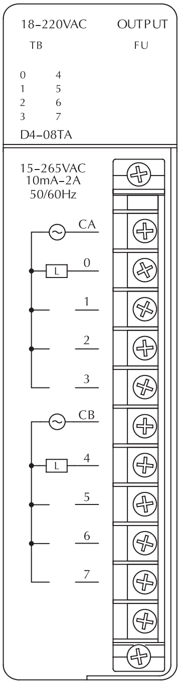
D4-08TA AC Output	
Outputs Per Module	8
Commons Per Module	2 (isolated)
Operating Voltage	15–265 VAC
Output Type	SSR Triac
Peak Voltage	265VAC
AC Frequency	47–63 Hz
On Voltage Drop	1.5 VAC @ 2A
Max Current	2A/point 5A/common @ 30°C 2A/common @ 60°C
Max Leakage Current	5mA @ 265VAC
Max Inrush Current	30mA for 10ms 10A for 100ms
Minimum Load	10mA
Base Power Required 5V	250mA max
OFF to ON Response	1ms
ON to OFF Response	1ms + 1/2 cycle
Terminal Type (Included)	Removable (D4-8IOCON)
Status Indicators	Logic Side
Fuses	1 (8A) per common, non-replaceable

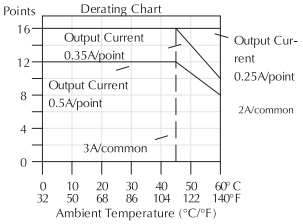
D4-08TA AC Output	
Outputs Per Module	16
Commons Per Module	2 (isolated)
Operating Voltage	15–265VAC
Output Type	SSR Triac
Peak Voltage	265VAC
AC Frequency	47–63 Hz
On Voltage Drop	1.5 VAC @ 0.5 A
Max Current	0.5 A/point 3A/common @ 45°C 2A/common @ 60°C
Max Leakage Current	4mA @ 265VAC
Max Inrush Current	15A for 10ms 10A for 100ms
Minimum Load	10mA
Base Power Required 5V	450mA max
OFF to ON Response	1ms
ON to OFF Response	1ms + 1/2 cycle
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side
Fuses	1 (5A) per common, non-replaceable

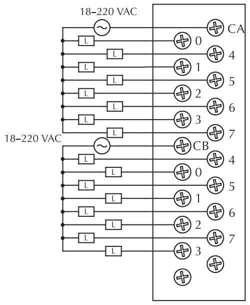


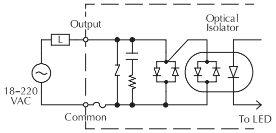


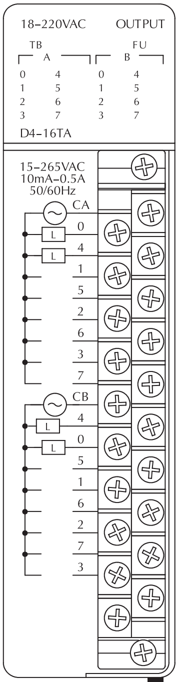






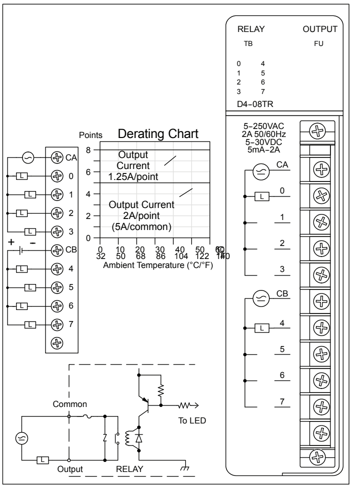






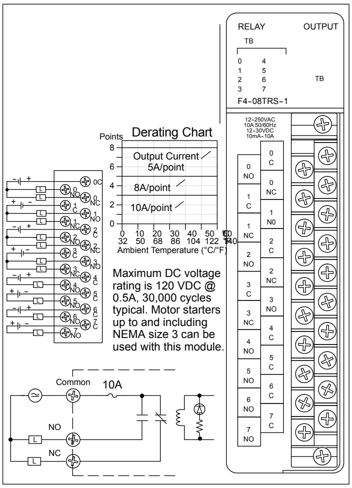
D4-08TR Relay Output	
Outputs Per Module	8 relays
Commons Per Module	2 (isolated)
Operating Voltage	5–30VDC / 5–250VAC
Output Type	Form A (SPST-N.O.)
Peak Voltage	30VDC / 256VAC
AC Frequency	47–63 Hz
On Voltage Drop	N/A
Max Current	2A/point 5A/common
Max Leakage Current	0.1 mA @ 265VAC
Max Inrush Current	2A
Minimum Load	5mA
Base Power Required 5V	550mA max
External DC Required	None
OFF to ON Response	12ms
ON to OFF Response	12ms
Terminal Type (Included)	Removable (D4-8IOCON)
Status Indicators	Logic Side
Fuses	1 (8A) per common Non-replaceable

Typical Relay Life (Operations)			
Maximum Resistive or Inductive Inrush Load Current	Operating Voltage		
	30 VDC	120 VAC	250 VAC
2A Resistive	100K	300K	200K
2A Inductive	100K	300K	60K
0.5 A Resistive	800K	1M	800K
0.5 A Inductive	300K	300K	200K



F4-08TRS-1 Relay Output	
Outputs Per Module	8 relays
Commons Per Module	8 (isolated)
Operating Voltage	12–30 VDC / 12–125 VAC *125–250 VAC
Output Type	4 Form C (SPST) 4, Form A (SPST N.O.)
Peak Voltage	30VDC / 250VAC @ 10A
AC Frequency	47–63 Hz
On Voltage Drop	N/A
Max Current	10A/point; 40A/module
Max Leakage Current	N/A
Max Inrush Current	10A
Minimum Load	100mA @ 12VDC
Base Power Required 5V	575mA max
External DC Required	None
OFF to ON Response	7ms
ON to OFF Response	9ms
Terminal Type (Included)	Removable (D4-8IOCON)
Status Indicators	Logic Side
Fuses	1 (10A) per common Non-replaceable

Typical Relay Life (Operations)			
Maximum Resistive or Inductive Inrush Load Current	Operating Voltage		
	28 VDC	120 VAC	250 VAC
1/4 HP	–	25K	–
10A	50K	50K	–
5A	200K	100K	–
3A	325K	125K	50K
0.05 A	>50M	–	–



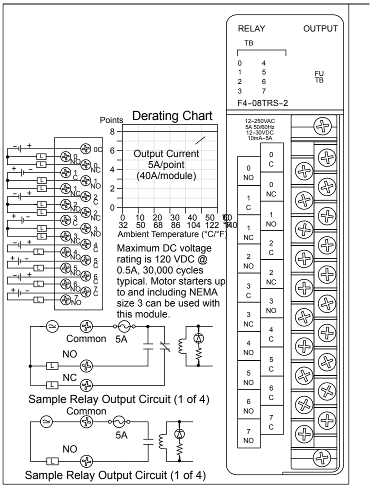
Maximum DC voltage rating is 120 VDC @ 0.5A, 30,000 cycles typical. Motor starters up to and including NEMA size 3 can be used with this module.

F4-08TRS-2 Relay Output	
Outputs Per Module	8 relays
Commons Per Module	8 (isolated)
Operating Voltage	12–30 VDC / 12–250 VAC
Output Type	4 Form C (SPST) 4, Form A (SPST N.O.)
Peak Voltage	30VDC 250VAC @ 5A
AC Frequency	47–63 Hz
On Voltage Drop	N/A
Max Current (Resistive)	5A/point 40A/module
Max Leakage Current	N/A
Max Inrush Current	10A
Minimum Load	5mA
Base Power Required 5V	1000mA max, 60mA/point
External DC Required	None
OFF to ON Response	7ms
ON to OFF Response	9ms
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side
Fuses	1 (10A 250V) per common User-replaceable part # D4-FUSE-2

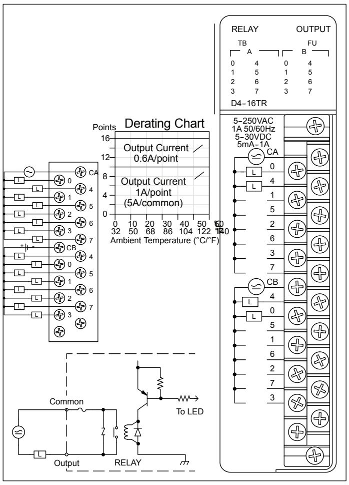
D4-16TR Relay Output	
Outputs Per Module	16 relays
Commons Per Module	2 (isolated)
Operating Voltage	5–30 VDC / 5–250 VAC
Output Type	Form A (SPST N.O.)
Peak Voltage	30VDC / 250VAC
AC Frequency	47–63 Hz
On Voltage Drop	N/A
Max Current (Resistive)	1A/point 5A/common
Max Leakage Current	0.1 mA @ 265VAC
Max Inrush Current	4A
Minimum Load	5mA
Base Power Required 5V	1000mA max, 60mA/point
External DC Required	None
OFF to ON Response	10ms
ON to OFF Response	10ms
Terminal Type (Included)	Removable (D4-16IOCON)
Status Indicators	Logic Side
Fuses	1 (18) per common Non-replaceable

Typical Relay Life (Operations)			
Maximum Resistive or Inductive Inrush Load Current	Operating Voltage		
	28 VDC	120 VAC	2450 VAC
5A	200K	100K	–
3A	325K	125K	50K
0.05 A	>50M	–	–

Typical Relay Life (Operations)			
Maximum Resistive or Inductive Inrush Load Current	Operating Voltage		
	30 VDC	120 VAC	250 VAC
1A Resistive	>1M	500K	300K
1A Inductive	400K	200K	100K
0.5 A Resistive	>2M	800K	500K
0.5 A Inductive	>1M	300K	200K



Maximum DC voltage rating is 120 VDC @ 0.5A, 30,000 cycles typical. Motor starters up to and including NEMA size 3 can be used with this module.



CPU SPECIFICATIONS AND OPERATION

CHAPTER 3

In This Chapter...

CPU Overview

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

- The differences between the various models of CPUs
- The steps required to setup and install the CPU

General CPU Features

The D4-454 and D4-454DC-1 are modular CPUs which can be installed in, 4, 6, or 8 slot bases. All I/O modules in the DL405 family will work with both of the CPUs. The D4-454 CPUs offer a wide range of processing power and program instructions. Both offer RLL and Stage program instructions (See Chapter 5). Both CPUs have extensive internal diagnostics that can be monitored from the application program or from an operator interface.

CPU Features

The D4-454 has a maximum 30.8K program memory comprised of 15.5K of ladder memory and 15.3K of V-memory (data registers). It supports a maximum of 3584 points of local and local expansion I/O, and 4096 points of remote I/O. It includes an additional internal ARM Cortex M3/4 micro-controller for greater processing power. The D4-454 has 210 instructions, including drum timers, print function, floating point math, trigonometric functions and PID loop control for 16 loops.

The D4-454 has a total of four built-in communication ports. Port 0 has a RS232 interface and supports K Sequence, DirectNET (hex only, Slave only) protocols. Port 1 is RS232/RS422 interface and supports K Sequence, DirectNET, Non-sequence and MODBUS protocols. Port 2 has RS232 interface and supports DirectNET, K-Sequence, Non-sequence, and Modbus RTU protocols. Port 3 is RS422 interface and supports K-Sequence, DirectNet, Non-sequence and MODBUS protocols.

CPU Electrical Specifications

Specification	D4-454	D4-454DC-1
Input Voltage, Nominal	120 or 240 VAC	24 VDC
Input Voltage Range	85–132 VAC and 170–240 VAC	20–29 VDC
Input Voltage Ripple	N/A	< 10%
Inrush Current, maximum	20A	10A
Power Consumption, maximum	50 VA	38W
Voltage Withstand (dielectric strength)	1 minute at 1500 VAC between primary, secondary, field ground and run relay	
Insulation Resistance	> 10MΩ at 500 VDC	
Output Voltage, auxiliary power supply	20–28 VDC (24 nominal), ripple more than 1V P-P	
Output Current, auxiliary power supply	24 VDC @ 400 mA maximum	

CPU Specifications

General Specifications

Feature	D4-454
Total Program Memory (words)	46.8K
Ladder Memory (words) built-in	31.5K
V-memory (words)	15.3K
Non-volatile V Memory (words)	No
Runtime Edit	Yes
Supports Override	Yes
RLL and RLLPLUS Programming	Yes
Direct SOFT (Programming for Windows)	Yes version 6.1 or later
Built-in Communication Ports	1 RJ12 RS232 1 15-pin RS232 1 25-pin RS232/RS422 1 25-pin RS422
Built-in Memory	Yes (M-RAM)
Number of Instructions Available	210
Control Relays	2048
Special Relays (system defined)	512
Stages in RLLPLUS	1024
V-memory (words)	15360
Timers	256
Counters	256
Immediate I/O	Yes
Interrupt Input	16 points
Subroutines	Yes
Drum Timers	Yes
Table Instructions	Yes
For/Next Loops	Yes
Math	Integer, Floating Point
ASCII	Yes
PID Loop Control, Built -in	Yes, 16 Loops
Time of Day Clock/Calendar	Yes
Internal Diagnostics	Yes
Password Security	Yes, multi-level
System Error Log	Yes
User Error Log	Yes
Battery Backup	D2-BAT-1 included

Local I/O and Local Expansion I/O

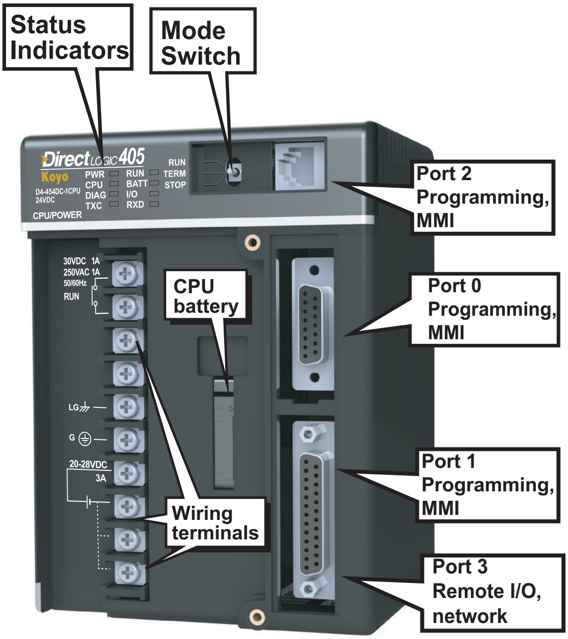
Feature	D4-454
Available Bases	4 (CPU Base, 3 Expansion Bases)
CPU Base Total I/O	512
Local Expansion I/O per Base	512
Total Local I/O and Local Expansion I/O	2048
Local Addressable Discrete Input Points	1024
Local Addressable Discrete Output Points	1024
Local Addressable Analog Input Channels	512
Local Addressable Analog Output Channels	512
Slots per Base	4/6/8
Discrete I/O Module Point Density	8/12/16/32/64

Serial Remote I/O

Feature	D4-454
Remote I/O Channels	3 (two D4-RM, vvvCPU Port 3)
Total Remote I/O	1536
Remote I/O per Channel	512
Remote I/O Slaves per Channel	7 (share 512 I/O points)
Remote I/O Distance	3300 ft (1000m)
Slots per Base	4/6/8
Discrete I/O Module Point Density	8/12/16/32/64

Ethernet Remote I/O

Feature	D4-454
Channels per CPU using (H4-ERM100)	Limited by power budget
Slaves per Ethernet Remote Master	16 (H4-EBC)
Total Ethernet Remote I/O points	16384
Total Bases per Slave (H4-EBC)	4 (H4-EBC base, three D4-EX bases)
Slots per Base	4/6/8
Discrete I/O Module Point Density	8/12/16/32/64



Toggle Switch Functions

The CPU mode switch on the D4-454 CPUs provides modes 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, DirectSOFT programming software or operator interface. Programs may be viewed or monitored but no changes may be made. If the mode 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.

Mode Switch	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 Debug modes are available. Mode and program changes are allowed by the programming / monitoring device.
STOP (Stop Program)	CPU is forced into the STOP mode. No changes are allowed by the programming / monitoring device.

There are two ways to change the CPU mode.

- 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.

Status Indicators

The status indicator LEDs on the CPU front panels have specific functions which 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
	Flashing	CPU is in Firmware Upgrade Mode
CPU	ON	CPU self diagnostics error
	OFF	CPU self diagnostics good
BATT **Note: Refer to page 3-8**	ON	Low battery voltage (V7745.12 must be On)
	OFF	CPU battery voltage is good or disabled
DIAG	ON	CPU self diagnostics or local bus error
	OFF	CPU self diagnostics and local bus good
I/O	ON	I/O self diagnostics error
	OFF	I/O self diagnostics good
TXD	ON	Data is being transmitted by the CPU
	OFF	No data is being transmitted by the CPU
RXD	ON	Data is being received by the CPU
	OFF	No data is being received by the CPU

Communication Ports

The D4-454 CPUs provide four Serial communication ports.

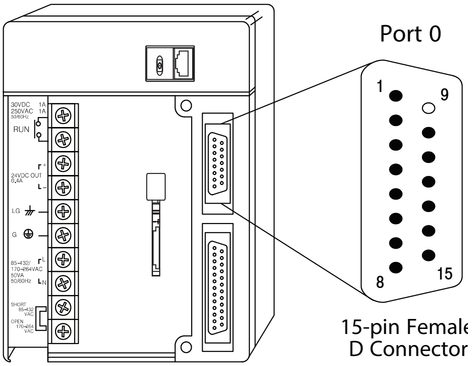
Port 0 Specifications

The first port is located on the 15-pin D-shell connector. It is for general programming such as DirectSOFT, or operator interface connections. The operating parameters for Port 0 are permanently set to the values shown.

- 15-Pin Female D shell connector
- Protocols: K Sequence, DirectNET (Hex only, Slave only)
- RS232, non isolated, distance with 15m (approx. 50ft)
- 9600 baud, 8 data bits, 1 start, 1 stop bit, Odd parity
- Asynchronous, half–duplex, DTE
- Supports Firmware updates

Port 0 Pin Descriptions

1	YOP	Sense connection between HPP and CPU
2	TXD	Transmit Data (RS232)
3	RXD	Receive Data (RS232)
4	ON-LINE	Request Communication (TTL)
5	ABNO	CPU error (TTL)
6	PRDY	CPU ready to communicate (TTL)
7	CTS	Clear to Send (RS232)
8	YOM	Sense connection between HPP and CPU
9	-	Not Used
10	LCBL	Sense cable connection (TTL)
11	5V2	5VDC for HPP logic
12	5V2	5VDC for LCD back-light
13	0V	Logic Ground
14	0V	Logic Ground
15	0V	Logic Ground

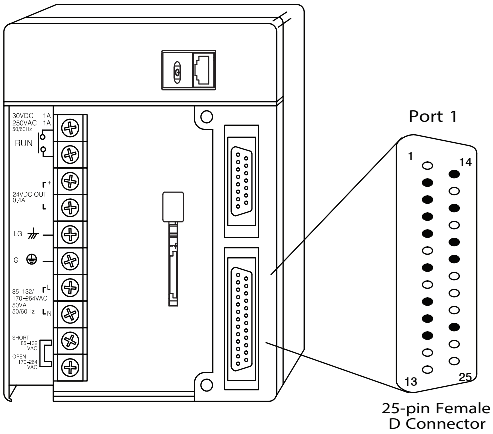


Port 1 Specifications

Port 1 is located on the 25-pin female D-shell connector. It is for general programming such as DirectSOFT, operator interfaces and networking. Port 1 provides additional features such as programmable baud rate, parity, ASCII/Hex mode and station addresses. Its RS422 signals support multi-drop networking and programming applications.

RS232 or RS422 is selected by cabling to the proper signal pin sets on the connector. Parity, ASCII/Hex mode and station address are selected by using DirectSOFT programming software.

- Protocols: K-Sequence, DirectNet, (master or slave only), Non-Sequence and MODBUS RTU (master or slave)
- RS232 / RS422, selectable address 1-90
- 2400 / 4800 / 9600 / 19200 / 38400 baud (default is 19200)
- 8 data bits, 1 start, 1 stop bit, Odd, Even or No parity (default is 8 data bits, 1 stop bit, Odd parity)
- Asynchronous, half–duplex, DTE
- Supports Firmware updates



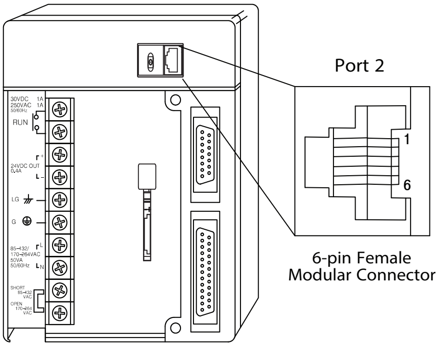
Port 1 Pin Descriptions		
1	–	Not Used
2	TXD	Transmit Data (RS232)
3	RXD	Receive Data (RS232)
4	RTS	Ready to Send (RS232)
5	CTS	Clear to Send (RS232)
6	–	Not Used
7	SG	Signal Ground (RS232/RS422)
8	–	(Port 3)
9	RXD+	Receive Data + (RS422)
10	RXD-	Receive Data – (RS422)
11	CTS+	Clear to Send + (RS422)
12	–	(Port 3)
13	–	(Port 3)
14	TXD+	Transmit Data –(RS422)
15	–	Not Used
16	TXD-	Transmit Data–(RS422)
17	–	Not Used
18	RTS-	Request to Send–(RS422)
19	RTS+	Request to Send + (RS422)
20	–	Not Used
21	–	Not Used
22	–	Not Used
23	CTS-	Clear to Send–(RS422)
24	–	(Port 3)
25	–	(Port 3)

Port 2 Specifications

The operating parameters for Port 2 on the D4-454 are configurable using DirectSOFT programming software on a programming device.

- 6-Pin Female modular (RJ12 phone jack) type connector
- Protocols: K-Sequence, DirectNET (master or slave only), Non sequence, Modbus RTU (master or slave)
- RS232, 2400 / 4800 / 9600 (default) / 19200 / 38400 baud
- Nodes (1–90)
- 8 data bits, one start, one stop; Odd, Even, or No parity; (default is 8 data bits, one stop; Odd parity)
- Supports Firmware updates

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



Port 2 Pin Descriptions

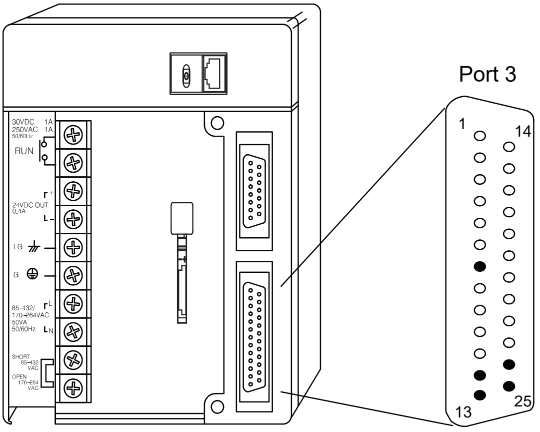
1	0V	Power (–) connection (GND)
2	5V	Power (+) connection
3	RXD	Receive Data (RS232)
4	TXD	Transmit Data (RS232)
5	5V	Power (+) connection
6	0V	Power (–) connection (GND)

Port 3 Specifications

The operating parameters for Port 3 on the D4-454 are configurable using DirectSOFT on a programming device.

- 25 Pin Female modular D type connector
- Protocols: K-Sequence, DirectNET (master or slave) Non-sequence, Modbus RTU (master or slave)
- RS422, selectable address 1-90, 2400 / 4800 / 9600 (default) / 19200 / 38400 baud
- Nodes (1–90)
- 8 data bits (default), one start, one stop (default); Odd (default), Even, or No parity
- Supports Firmware updates

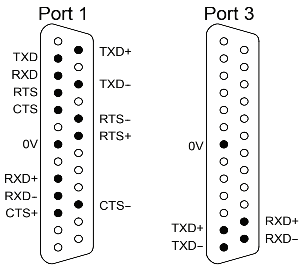
Port 3 Pin Descriptions		
1	–	Not Used
2	–	(port 1)
3	–	(port 1)
4	–	(port 1)
5	–	(port 1)
6	–	Not Used
7	SG	Signal Ground (RS422)
8	–	Not Used
9	–	(port 1)
10	–	(port 1)
11		(port 1)
12	TXD+	Transmit Data + (RS422)
13	TXD-	Transmit Data–(RS422)
14	–	(port 1)
15	–	Not Used
16	–	(port 1)
17	–	Not Used
18	–	(port 1)
19	—	(port 1)
20	–	Not Used
21	–	Not Used
22	–	Not Used
23	–	(port 1)
24	RXD+	Receive Data + (RS422)
25	RXD-	Receive Data–(RS422)



25-pin Female
D Connector

A drawing summarizing the pin locations and functions of ports 1 and 3 on the 25-pin connector is to the right. The two logical ports share two ground pins, but have separate communications data pins. When using both logical ports, you will probably have to make a custom connector which divides the signals in two for two separate cables.

Two Logical Ports on
the 25 Pin Connector



Using Battery Backup

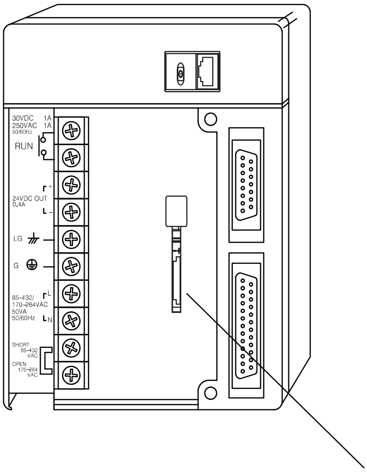
A lithium battery is available to maintain the system RAM retentive memory when the system is without external power. Typical CPU battery life is five years, which includes PLC runtime and normal shutdown periods. However, consider installing a fresh battery if your battery has not been changed recently and the system will be shutdown for a period of more than ten days. The battery indicator will flash on and off at 2 Hz when a battery needs changing.

NOTE: *Be sure to back up your V-memory and system parameters before replacing your CPU battery. You can do this by using DirectSOFT version 6.1 or later, to save the project, V-memory and system parameters to a personal computer. (File > Save Project). To prevent memory loss, the CPU battery can be changed while the system is powered up. If the CPU has been powered off you should power-up the CPU for at least 5 seconds prior to changing the battery. This ensures the capacitor used to maintain the necessary voltage levels for retaining memory is fully charged.*

To install the D2-BAT-1 battery in the CPU:

- Press the retaining clip on the battery door and swing the battery door open (swings downward).
- Place the battery into the coin-type slot with the (+) or larger side out.
- Close the battery door making sure that it locks securely in place.
- 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.



The battery backup is available immediately after the battery has been installed. The CPU indicator will blink if the battery is low (refer to table on page 3-5). Special Relay 43 (SP43) will also be set when the battery is enabled by setting bit 12 of V7633 (V7633.12). 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 V7633.12. The battery will do the same, but for a much longer time.

CPU Setup Information

Setting the Clock and Calendar

The D4-454 CPUs have a Clock / Calendar that can be used for many purposes. With DirectSOFT programming software you will use the PLC Setup menu options. There are two instructions that allow you change or modify the time and date from within the application program. Chapter 5 provides information on the DATE and TIME instructions that are used to establish the clock and calendar information.

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

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

Variable/Fixed Scan Time Feature

The D4-454 CPU offers three types of scan time configurations:

- Variable–this is the standard scan time setting, in which the PLC scan is running as fast as the ladder program execution allows.
- Fixed–the scan time may be set to be constant, from 10ms to 9999ms. The operating system inserts a delay after each ladder scan to accomplish the requested fixed scan.
- Time–The PLC operates with a variable scan, but generates a watchdog timeout error if the scan exceeds the specified amount. You can use this to trap program execution errors, for example.

To select the desired D4-454 scan time option, use DirectSOFT and go online with the D4-454. Then select the PLC > Diagnostics > Scan Time > Setup. The three choices of Variable, Fixed, or Time appear.

Password Protection

The D4-454 CPUs have a password protection function using DirectSOFT. 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.)

Multilevel Password The D4-454 CPUs also feature an intermediate level of protection that you can choose by making the first character of the password the character "A". The remaining seven characters must be numeric (0–9). The intermediate password allows an Operator Interface to communicate with the processor, allowing V memory data changes and does NOT allow edits to the Ladder program.

Clearing an Existing Program

Before entering a new program, it's a good idea to always clear ladder memory. You can clear an existing program from the CPU and once you are connected using the DirectSOFT programming software go to PLC > Clear Memory.

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 logic project) removed. Is is the policy of AutomationDirect to require the memory of the PLC to be cleared along with the password.

Initializing System Memory

The D4-454 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.

To initialize the “scratchpad”, you need to be connected with the D4-454, go to PLC > Setup > Initialize Scratchpad.

WARNING: You may never have to use this feature unless you want to clear any setup information that is stored in 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.

Setting Retentive Memory Ranges

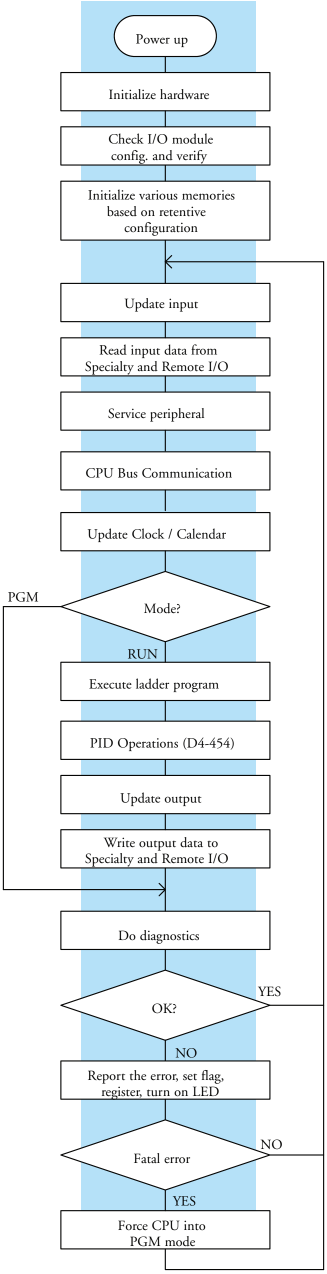
The D4-454 CPUs provide certain ranges of retentive memory by default. The default ranges are suitable for many applications, but you can change them if your application requires additional retentive ranges or no retentive ranges at all. The default settings are:

Memory Area	D4-454	
	Default Range	Available Range
Control Relays	C1000 – C3777	C0 – C3777
V-Memory	V400 – V37777	V0 – V37777
Timers	None by default	T0 – T377
Counters	CT0 – CT377	CT0 – CT377
Stages	None by default	S0 – S1777

CPU Operation

Achieving the proper control for your equipment or process requires a good understanding of how D4-454 CPUs control all aspects of system operation. The flow chart 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 Debug Mode.
- CPU Timing — The two important areas we discuss are the I/O response time and the CPU scan time.
- CPU Memory Map — The CPUs memory map shows the CPU addresses of various system resources, such as timers, counters, inputs, and outputs.


Power up
Initialize hardware
Check I/O module config. and verify
Initialize various memories based on retentive configuration
Update input
Read input data from Specialty and Remote I/O
Service peripheral
CPU Bus Communication
Update Clock / Calendar
PGM
Mode?
RUN
Execute ladder program
PID Operations (D4-454)
Update output
Write output data to Specialty and Remote I/O
Do diagnostics
OK?
YES
NO
Report the error, set flag, register, turn on LED
Fatal error
NO
YES
Force CPU into PGM mode

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.

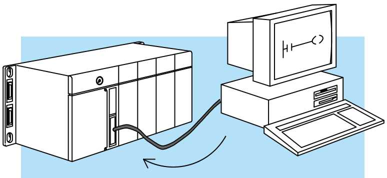
In Run Mode, the CPU executes the user ladder program. Immediately afterwards, any PID loops which are configured are executed. Then the CPU writes the output results of these two tasks to the appropriate output points.

Error detection has two levels. 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.

You can use the mode switch on the CPU to select Program Mode operation.



Download Program

Run Mode Operation

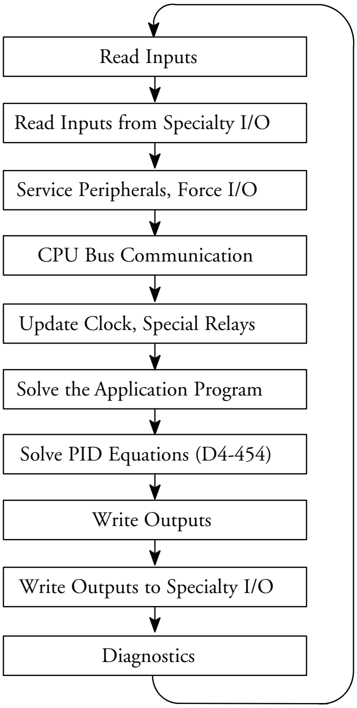
In Run Mode, the CPU executes the application program, does PID calculations for configured PID loops, and updates the I/O system. You can perform many operations during Run Mode. Some of these include:

- 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.

You can also edit the program during Run Mode. The Run Mode Edits are not "bump-less." 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.


Read Inputs
Read Inputs from Specialty I/O
Service Peripherals, Force I/O
CPU Bus Communication
Update Clock, Special Relays
Solve the Application Program
Solve PID Equations (D4-454)
Write Outputs
Write Outputs to Specialty I/O
Diagnostics

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 Five.

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 High Speed Counter 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. There are two basic types of forcing available with the D4-454 CPUs:

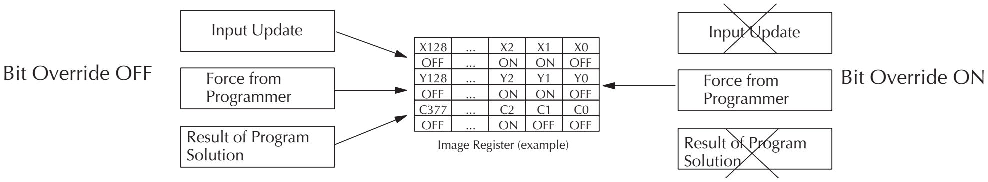
- Forcing from a peripheral – not a permanent force, good only for one scan
- Bit Override – holds the I/O point (or other bit) in the current state. Valid bits are X, Y, C, T, CT, and S. (These memory types are discussed in more detail later in this chapter).

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 — 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 DirectSOFT. 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.



X128	...	X2	X1	X0
OFF	...	ON	ON	OFF
Y128	...	Y2	Y1	Y0
OFF	...	ON	ON	OFF
C377	...	C2	C1	C0
OFF	...	ON	OFF	OFF

Image Register (example)

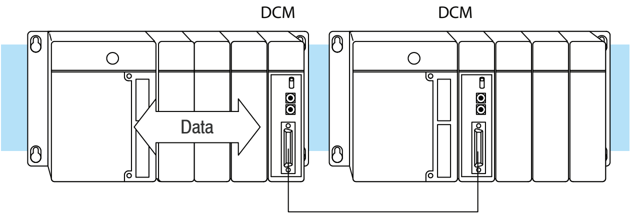
WARNING: Only authorized personnel fully familiar with all aspects of the application should make changes to the program. Changes during Run Mode become effective immediately. Make sure you thoroughly consider the impact of any changes to minimize the risk of personal injury or damage to equipment.

Update Special Relays and Special Registers

There are certain V-memory locations that contain register information. This portion of the execution cycle makes sure these locations get updated on every scan. Also, there are several different Special Relays, such as diagnostic relays, etc., that are also updated during this segment.

CPU Bus Communication

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



Update Clock, Special Relays and Special Registers

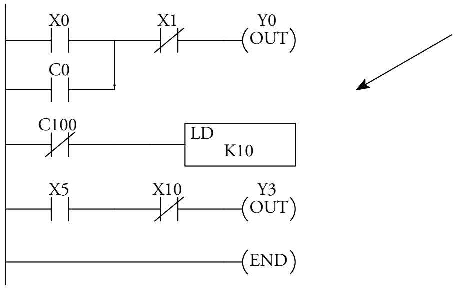
The D4-454 CPUs have an internal real-time clock and calendar timer which is accessible to the application program. Special V-memory locations hold this information. This portion of the execution cycle makes sure these locations get updated on every scan. Also, there are several different Special Relays, such as diagnostic relays, etc., that are also updated during this segment.

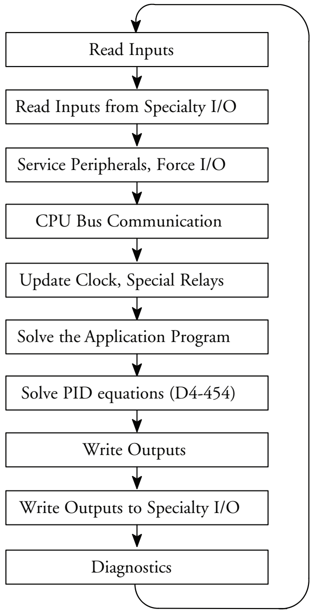
Solve Application Program

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 desired output responses.

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 CPU uses the output image register area to store the status of the desired action for the outputs. Output image register locations are designated with a Y followed by a memory location. The actual outputs are updated during the write outputs segment of the scan cycle. There are immediate output instructions available that will update the output points immediately instead of waiting until the write output segment. A complete list of the Immediate instructions is provided in Chapter 5.




Read Inputs
Read Inputs from Specialty I/O
Service Peripherals, Force I/O
CPU Bus Communication
Update Clock, Special Relays
Solve the Application Program
Solve PID equations (D4-454)
Write Outputs
Write Outputs to Specialty I/O
Diagnostics

The internal control relays (C), the stages (S), the global relays (GY), 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.

Solve PID Loop Equations

The D4-454 CPUs can process up to 16 PID loops. The loop calculations are run as a separate task from the ladder program execution, immediately following it. Only loops which 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 which 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 updated 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.*

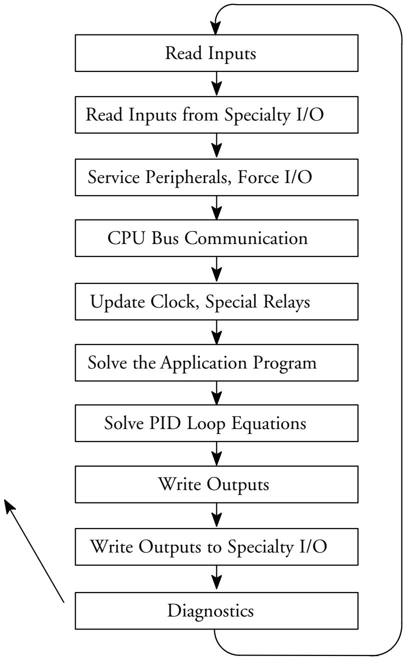
Diagnostics

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

D4-454 CPUs automatically detect and report many different error conditions. Please refer to the Error Codes Appendix which contains a listing of the various error codes available with the DL405 system.

One of the more important diagnostic tasks is the scan time calculation and watchdog timer control. D4-454 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.


Read Inputs
Read Inputs from Specialty I/O
Service Peripherals, Force I/O
CPU Bus Communication
Update Clock, Special Relays
Solve the Application Program
Solve PID Loop Equations
Write Outputs
Write Outputs to Specialty I/O
Diagnostics

I/O Response Time

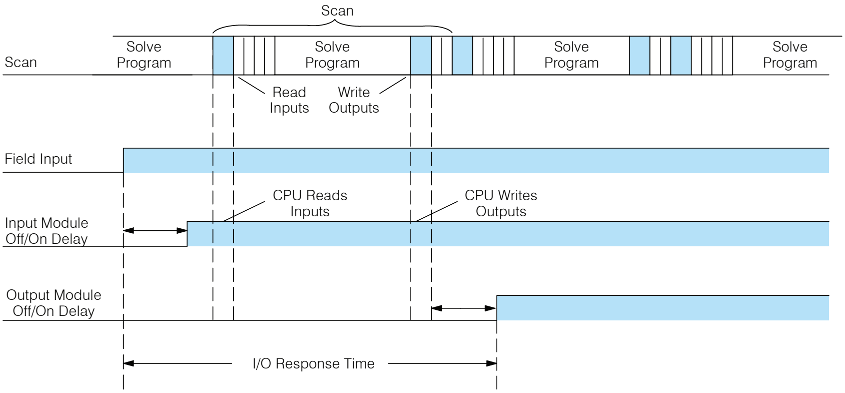
Is Timing Important for Your Application

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. There are four things that 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 just 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:

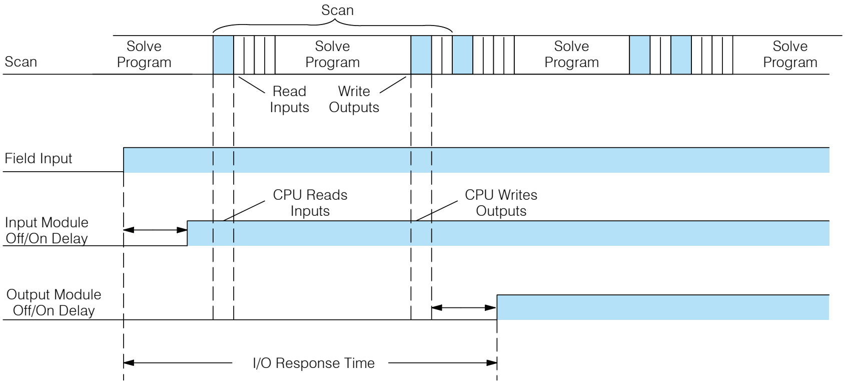
Input Delay + Scan Time + Output Delay = Response Time

Normal Minimum I/O Response

The I/O response time is longest when the modules senses the input change just 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

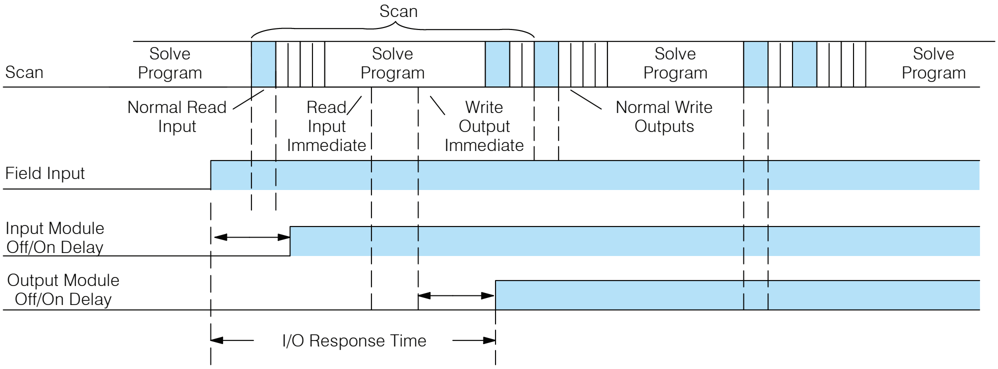


Improving Response Time

There are a few things you can do 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

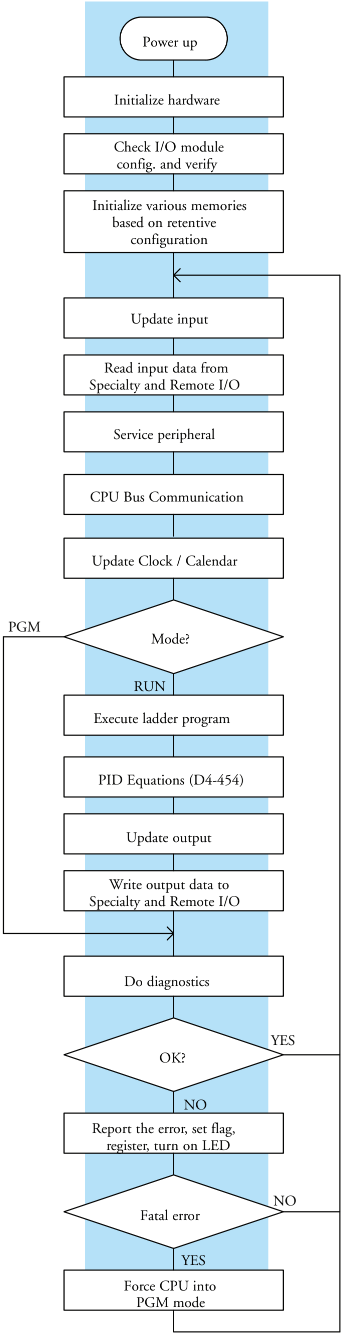
The scan time covers all the cyclical tasks that are performed by the operating system. You can use DirectSOFT to display the minimum, maximum, and current scan times that have occurred since the previous Program Mode to Run Mode transition. This information can be very important when evaluating the performance of a system.

As we've shown previously there are several segments that make up the scan cycle. Each of these segments require 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, these things are usually dictated by the application.

For example, if you have a need to count pulses at high rates of speed, then you'll 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's much faster and cheaper to install a single remote I/O cable than it is to run all those wires for each individual I/O point.

The following paragraphs provide some of the general information on how much time some of the segments can require.


Power up
Initialize hardware
Check I/O module config. and verify
Initialize various memories based on retentive configuration
Update input
Read input data from Specialty and Remote I/O
Service peripheral
CPU Bus Communication
Update Clock / Calendar
PGM
Mode?
RUN
Execute ladder program
PID Equations (D4-454)
Update output
Write output data to Specialty and Remote I/O
Do diagnostics
OK?
YES
NO
Report the error, set flag, register, turn on LED
Fatal error
NO
YES
Force CPU into PGM mode

Initialization Process

The CPU performs an initialization task once the system power is on. The required time depends on system loading, such as the number of I/O modules installed. The initialization task is performed once at power-up, so it does not affect the scan time for the application program.

Initialization	
Minimum Time	1.9 Seconds
Maximum Time	3.3 Seconds

Reading Inputs

The time required to read the input status for the local and expansion input modules depends on the number of input points in the bases, and the number of input modules being used. The following table shows typical update times.

Timing Factors	
Overhead	20.0 µs
Per input modules	13.0 µs
Per input point	6.3 µs

Update Clock/Calendar, Special Relays, Special Registers

For example, the time required for a D4-454 to read two 16-point input modules would be calculated as follows (Where NM is the number of modules and NI is the total number of input points.)

- Formula
- Time = 20 µs + (13 µs x NM) + (6.3 µs x NI)

Example

- Time = 20 µs + (13 µs x 2) + (6.3 µs x 16)
- Time = 146.8 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 the number of modules and the number of input points.

For example, the time required for D4-454 to read two 32-point input modules (located in a Remote base) and the input points associated with a single High-Speed Counter module would be calculated as follows. (Where NM is the number of modules and NI is the number of input points in a module.)

Specialty Module	
Overhead	20.0 µs
Per Module (with inputs)	13.0 µs
Per input point	13.8 µs

Remote Module	
Overhead	19.0 µs
Per Module (with inputs)	62.0 µs
Per input point	11.2 µs

Remote I/O

- Formula
- Time = 19 µs + (62 µs x NM) + (11.2 µs x NI)

Example

- Time = 19µs + (62 µs x 2) + (11.2 µs x 32)
- Time = 501.4 µs

High Speed Counter

- Formula
- Time = 20 µs + (13 µs x NM) + (13.8 µs x NI)

Example

- Time = 20 µs + (13 µs x 2) + (13.8 µs x 16)
- Time = 266.8 µs

Total Time = 501.4 µs + 266.8 µs = 768.2 µs

Service Peripherals

Communication request 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)		D4-454
Nothing Connected	Min. & Max.	0 µs
Port 0	Send Min. / Max.	38 / 38 µs
	Rec. Min. / Max.	45 /45 µs
Port 1	Send Min. / Max.	41 / 48 µs
	Rec. Min. / Max.	47 / 59 µs
Port 2	Send Min. / Max.	41 / 48 µs
	Rec. Min. / Max.	47 / 59 µs
Port 3	Send Min. / Max.	38 / 38 µs
	Rec. Min. / Max.	45 / 45 µs

To Service Request	D4-454
Minimum	96 µs
Run Mode Max.	160 ms
Program Mode Max.	11.2 Second

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.

CPU Bus Communications

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		D4-454
Program Mode	Minimum	12.0 µs
	Maximum	12.0 µs
Run Mode	Minimum	22.0 µs
	Maximum	29.0 µs

Writing Outputs

The time required to write the output status for the local and expansion I/O modules depends on the number of output points that are in these bases and the number of output modules being used. The following table shows typical update times required by the CPU.

Timing Factors	D4-454
Overhead	15.0 µs
Per output module	13.0 µs
Per output point	14.1 µs

For example, the time required for a D4-454 to write data for two 32-point output modules would be calculated as follows (where NM is the number of modules and NO is the number of output points in a module.

- Formula
- Time = 15µs + (13µs x NM) + (14.1 µs x NI)

Example

- Time = 15µs + (13µs x 2) + (14.1 µs x 32)
- Time = 492.2 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 the number of modules and the number of output points.

Specialty Modules	D4-454
Overhead	18.0 µs
Per module (with outputs)	13.0 µs
Per output point	14.1 µs
Remote Modules	D4-454
Overhead	15.0 µs
Per module (with outputs)	54.0 µs
Per output point	13.9 µs

For example, the time required for D4-454 to write two 32-point output modules (located in a Remote base) and the output points associated with a single High-Speed Counter module would be calculated as follows. (Where NM is the number of modules and NI is the number of output points in a module.

Remote I/O

- Formula
- Time = 15µs + (54µs x NM) + (13.9 µs x NI)

Example

- Time =15µs + (54µs x 2) + (13.9 µs x 32)
- Time = 567.8 µs

High Speed Counter

- Formula
- Time = 18µs + (13µs x NM) + (14.1 µs x NI)

Example

- Time = 18µs + (13µs x 2) + (14.1 µs x 16)
- Time = 269.6 µs

NOTE: *The 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 D4-454 CPU performs 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	D4-454
Minimum	282.0 µs
Maximum	398.0 µs

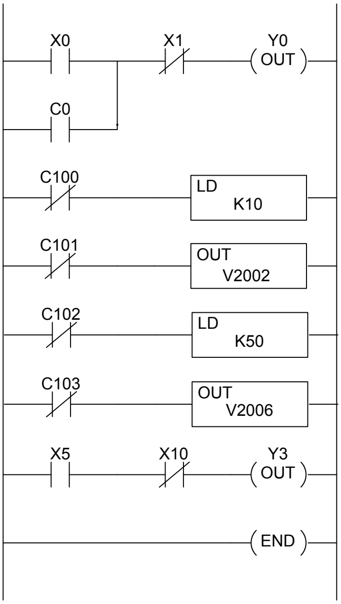
Application Program Execution

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 D4-454 running the program shown would be calculated as follows.

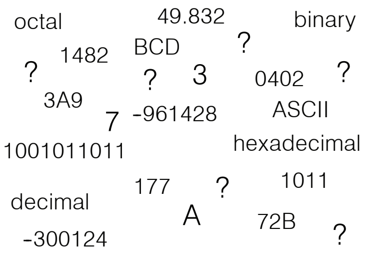


Program Control Instructions

Instruction	Time
STR X0	0.96 µs
OR C0	0.9 µs
ANDN X1	0.9 µs
OUT Y0	2.9 µs
STRN C100	1.0 µs
LD K10	12.7 µs
STRN C101	1.0 µs
OUT V2002	4.7 µs
STRN C102	1.0 µs
LD K50	12.7 µs
STRN C103	1.0 µs
OUT V2006	4.7 us
STR X5	4.7 µs
ANDN X10	0.9 µs
OUT Y3	2.9 µs
END	8.5 µs
TOTAL	61.6 µs

The D4-454 offers 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

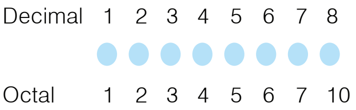


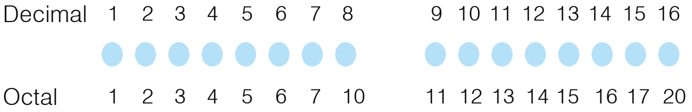
If you are a new PLC user or are using AutomationDirect PLCs for the first time please take a moment to study how our PLCs use numbers. You'll find that each PLC manufacturer has their own conventions on the use of numbers in their PLCs. We want to take just a moment to familiarize you with how numbers are used in the DirectLOGIC PLCs. As any good computer does, PLCs store and manipulate numbers in binary form: just ones and zeros. So why do we have to deal with numbers in so many different forms? Numbers have meaning, and some representation 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 the Number Systems Appendix for numbering system details.)

PLC Resources

PLCs offer a fixed amount 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 DirectLOGIC are counted in octal. It's easier for computers to count in groups of eight than ten, because eight is an even power of 2.

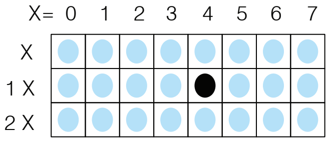
Octal means simply counting in groups of eight things at a time. In the figure to the right, there are eight circles. The quantity in decimal is "8", but in octal it is "10" (8 and 9 are not valid in octal). In octal, "10" means 1 group of 8 plus 0 (no individuals).




Decimal 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Octal 1 2 3 4 5 6 7 10 11 12 13 14 15 16 17 20

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 (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 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.

V-memory address (octal)	V-memory data (binary) MSB															LSB
V2017	0	1	0	0	1	1	1	0	0	0	1	0	1	0	0	1

A frequently-asked questions is “How do I tell if a number is binary, octal, BCD, or hex”? The answer is that we usually cannot tell just by looking at the data, but it does not really matter. What matters is: the source or mechanism which writes data into a V-memory location and the thing which later reads it must both use the same data type (i.e., octal, hex, binary, or whatever). The V-memory location is just 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 (10 fingers, 10 toes), 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 four binary bits (a nibble). This permits each V-memory location to store four 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 DirectSOFT allows us to enter and view data in BCD. Special RLL instructions convert from BCD to binary, or visa-versa.

BCD number	4	9	3	6
	8 4 2 1	8 4 2 1	8 4 2 1	8 4 2 1
V-memory storage	0 1 0 0	1 0 0 1	0 0 1 1	0 1 1 0

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.

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 just a convenient way for humans to view full binary data.

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

Hexadecimal number	A	7	F	4
V-memory storage	1 0 1 0	0 1 1 1	1 1 1 1	0 1 0 0

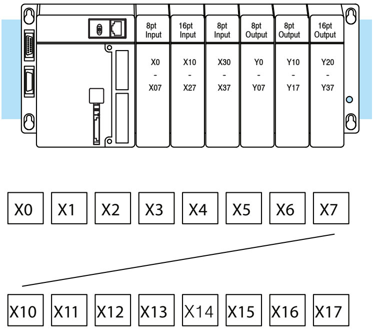
Memory Map

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, part 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 D4-454 CPUs. Memory maps follow the memory descriptions.

Octal Numbering Systems

All memory locations or areas are numbered in octal (base 8). 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.

Refer to the previous section on PLC Numbering Systems for more on octal numbering.

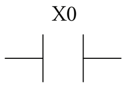


Discrete and Word Locations

As you examine the different memory types, you'll notice two types of memory in the DL405, 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, etc.

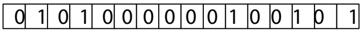
Some information is automatically stored in V-memory. For example, the timer current values are stored in V-memory.

Discrete – On or Off, 1 bit



V-Memory Locations for Discrete Memory Areas

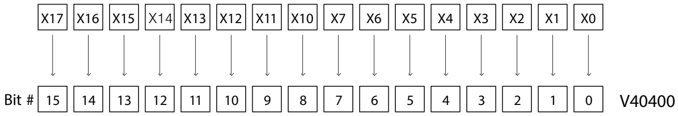
Word Locations – 16 bits



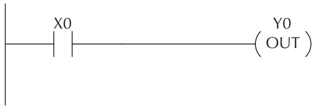
The discrete memory area is for inputs, outputs, control relays, special relays, stages, global relays, timer status bits and counter status bits. However, you can also access the bit data types as V-memory word. Each V-memory location contains 16 consecutive discrete locations. For example, the following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

These discrete memory areas and the corresponding V-memory locations are listed in the Memory Map tables for the D4-454 in this chapter.

16 Discrete (X) Input Points

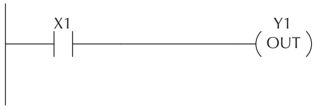


Input Points (X Data Type)



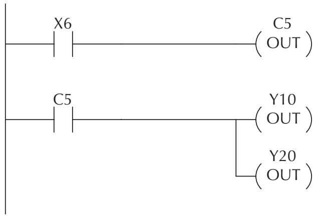
The discrete input points are noted by an X data type. Refer to the memory maps for the number of discrete input points for your CPU. In this example the output point Y0 will energize when input X0 turns on.

Output Points (Y Data Type)



The discrete output points are noted by a Y data type. Refer to the memory maps for the number of discrete input points for your CPU. In this example, output point Y1 will energize when input X1 turns on.

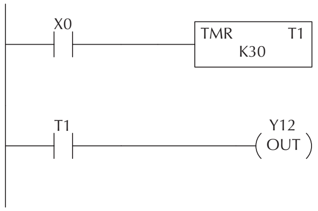
Control Relays (C Data Type)



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. They are internal to the CPU. Because of this, control relays 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 contains 16 consecutive discrete locations.

In this example, memory location C5 will energize when input X6 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)



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 bits will be on when the current value is equal or greater than the preset value of a corresponding timer.

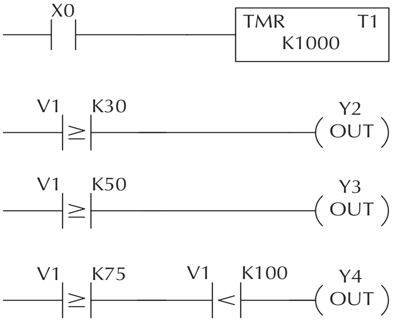
In this example, when input X0 turns on, timer T1 will start. When the timer reaches the preset of 3 seconds (K of 30) timer status contact T1 turns on. When T1 turns on, output Y12 turns on. Turning off X0 resets the timer.

***NOTE:** Some timers and counters use one V-memory register, and other types require two V-memory registers. See the instruction descriptions in Chapter 5.*

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.

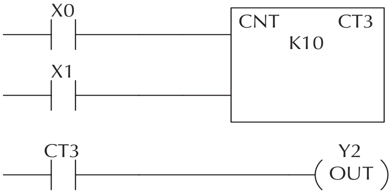
The primary reason for this is programming flexibility. The example shows you how you can use relational contacts to monitor several time intervals from a single timer.



Counters and Counter Status Bits (CT Data type)

There are 128 counters available in the CPU. 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.

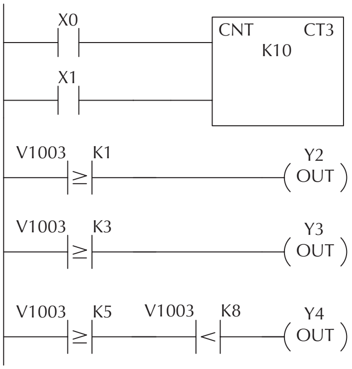
Each time contact X0 transitions from off to on, the counter increments by one. (If X1 comes on, the counter is reset to zero.) When the counter reaches the preset of 10 counts (K of 10) counter status contact CT3 turns on. When CT3 turns on, output Y2 turns on.



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 can also be designated as CTA0 (Counter Accumulated) for Counter 0 and CTA01 for Counter 1.

The primary reason for this is programming flexibility. The example shows how you can use relational contacts to monitor the counter values.

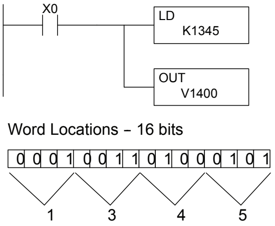


Word Memory (V Data Type)

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.

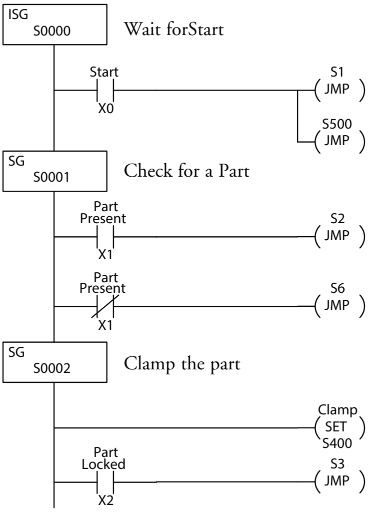


Stages (S Data type)

Stages are used in RLLPLUS 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 RLLPLUS 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.

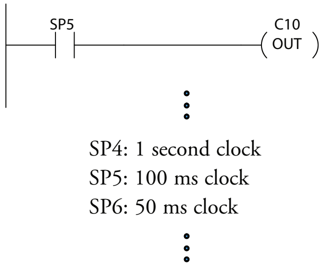
Ladder Representation



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. Please see the Special Relays Appendix for a complete listing of the special relays.

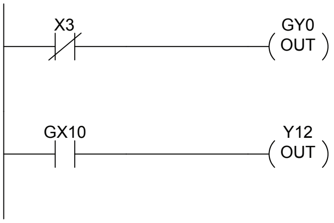
In this example, control relay C10 will energize for 50ms and de-energize for 50ms because SP5 is a pre-defined relay that will be on for 50ms and off for 50ms.



Remote I/O Points (GX and GY 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. There are setup routines that must be placed in your application program to designate which locations are inputs and which are outputs. (The DL405 Remote and Slice I/O modules manual provides the details.)

In this example, memory location GY0 turns on when local input X3 is not ON. On the second rung, local output Y12 will turn ON when GX10 turns on.



System Parameters (V Data Type)

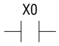
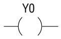
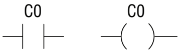
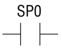
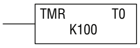
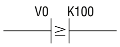
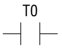
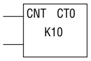
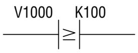
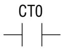
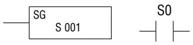
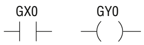
Many system parameters, such as error codes, are automatically stored in pre-defined V-memory locations. These memory locations store clock / calendar information, error codes and other types of system setup information.

System V-Memory	Description of Contents
V737	Contains a BCD value (from 3 to 999) for Timed-interrupt 17 feature.
V7633	Bit 12 enables the low battery warning indicator.
V7747	Contains a 10 mS calendar timer used with the Clock / Calendar
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. (0=Sun., 1=Mon, 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)

System V-Memory	Description of Contents
V736	Contains a BCD value (from 3 to 999) for Timed-interrupt 16 feature.
V7746	454: Battery voltage in tenths of a volt, (e.g., V7746 = 0031 is 3.1 Volts)

System V-Memory (continued)	Description of Contents
V7751	Fault Message Error Code — stores the 4-digit BCD code used with the FAULT instruction when the instruction is executed. If you've used ASCII messages, 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	Communications Error Code — stores the minor error code.
V7760	Module Error — identifies the base and slot number.
V7762	Module Error — identifies the type of error.
V7763	Program Grammatical Error — identifies the location of a syntax error in a program.
V7764	Program Grammatical Error — identifies the type of error.
V7765	Scan — stores the total number of scan cycles that have occurred since the last Program Mode to Run Mode transition.
V7775	Scan — stores the current scan time.
V7776	Scan — stores the minimum scan time that has occurred since the last Program-to-Run Mode transition.
V7777	Scan — stores the maximum scan time that has occurred since the last Program-to-Run Mode transition.

Systems 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	ON: The last state of inputs will be maintained OFF: The inputs will turn off when communication is lost
C743	Re-Start: Turning on this relay will resume after a communication hang-up on an error
C750 to C757	Setup Error: The corresponding relay will be ON if the setup table contains an error (C7250 = master, C751 =slave 1 C757 = slave 7)
C760 to C767	Communication Ready: The corresponding relay will be ON if the setup table data is valid (C760 = master, C761 = slave 1, C767 = slave 7)

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	V41100 – V41117	256	
Timer Current Values	None	V00000 – V00377	256	
Timer Status Bits	T0 – T377	V41100 – V41117	256	
Counters	CT0 – CT377	V41140 – V41157	256	
Counter Current Values	None	V01000 – V01377	256	
Counter Status Bits	CT0 – CT377	V41140 – V41157	256	
User Data Types	None	V1400 – V7377 V10000 –V36777	3072 11776	None specific, use with many instructions
Stages	S0 – S1777	V41000 –V41077	1024	
Remote In / Out	GX0 – GX3777 GY0 – GY3777	V40000 –V40177 V40200 –V40377	2048 2048	
System Parameters	None	V700 – V777 V7400 – V7777 V37000 –V37777	832	None specific, use with many instructions

DL405 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., which simplifies understanding the memory address. The use of 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 to reference memory locations.

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, it's 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, it's alias is VGX0. V40001 is the word memory reference for discrete bits GX20 through GX 37,therefore, its alias is VGX20.
V40200	VGY	V40200 is the word memory reference for discrete bits GY0 through-GY17, therefore, it's alias is VGY0. V40201 is the word memory reference for discrete bits GY20 through GY 37, therefore, it's alias is VGY20.
V40400	VX0	V40400 is the word memory reference for discrete bits X0 through X17, therefore, it's 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, it's 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, it's 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, it's 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, it's 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, it's 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, it's alias is VSP0. V41201 is the word memory reference for discrete bits SP20 through SP37,therefore, its alias is VSP20.

X Input / Y Output Bit Map

This table provides a listing of individual Input and Output points as associated with each V-memory address bit in the D4-454 CPUs.

MSB						D4-454 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
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
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

MSB						DL405 CPUs Additional Input (X) and Output (Y) Points (cont'd)									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								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
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	736	775	774	773	772	771	770	767	766	765	764	763	762	761	760	V40637

MSB							Additional 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	2414	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

MSB							Additional 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

MSB							Additional Control Relays (C) (cont'd)								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

Timer and Counter Status Bit Map

This table provides a listing of individual timer and counter contacts associated with each V-memory address bit.

MSB							Timer (T) and Counter (CT) Contacts								LSB	Timer Address	Counter 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	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
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 D4-454.

MSB							Additional Timer (T) Contacts								LSB	Timer Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
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 D4-454.

MSB							Additional Counter (CT) Contacts								LSB	Counter Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
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 individual remote I/O points associated with each V-memory address bit. The D4-454 uses the GX addresses for remote input point references and the GY addresses for remote ouput point references.

MSB					Remote I/O (GX) and (GY) Point										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	V40014	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	V40017	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

This portion of the table shows additional Remote I/O (GX) points and (GY) remote output available with the D4-454.

MSB				Additional Remote I/O (GX) Points											LSB	GX Address	GY 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	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

MSB	Additional 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		
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	2746	2745	2744	2743	2742	2741	2740	V40136	V40336
2777	2776	2775	2774	2773	2772	2771	2770	2767	2766	2765	2764	2763	2762	2761	2760	V40137	V40337

MSB	Additional 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		
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

Stage Control / Status Bit Map

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

MSB							Stage (S) Control Bits								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	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
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

MSB	Additional Stage (S) Control Bits (continued)														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

SYSTEM DESIGN AND CONFIGURATION

CHAPTER 4

In This Chapter...

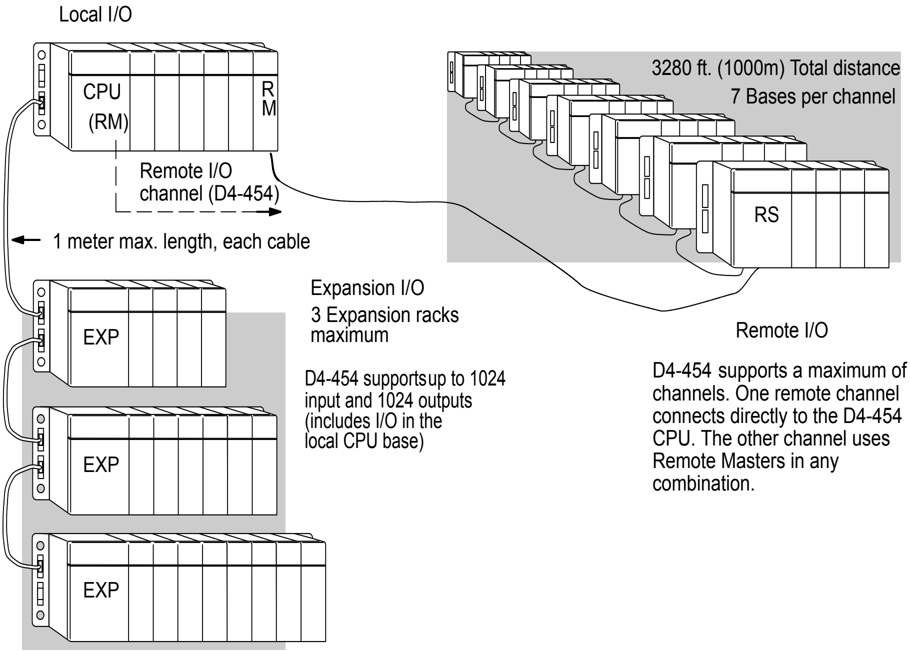
D4-454 System Design Strategies

I/O System Configurations

The D4-454 PLCs offer the following ways to add networking to the system:

- **Local I/O** - consists of I/O modules located in the same base as the CPU.
- **Expansion I/O** - consists of I/O modules in expansion bases located close the local base. Expansion cables connect them to the local CPU base's serial bus in daisy-chain fashion.
- **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 port 3 on the D4-454.

A D4-454 system can be developed using many different arrangements of these configurations. All I/O configurations use the standard complement of DL405 I/O modules and bases. Below is a brief description of each of these configurations. Examples of each configuration are discussed in detail later in this chapter.



D4-454 supportsup to 1024 input and 1024 outputs (includes I/O in the local CPU base)

D4-454 supports a maximum of 3 channels. One remote channel connects directly to the D4-454 CPU. The other channel uses Remote Masters in any combination.

Network Configurations

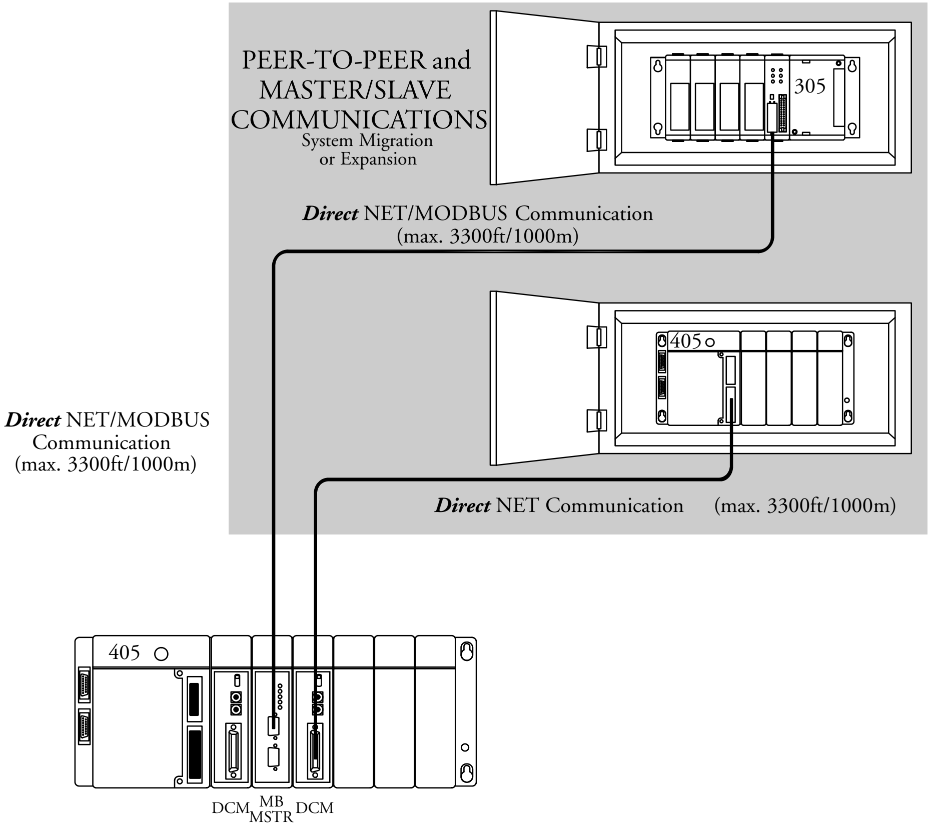
The D4-454 PLCs offer the following four ways to add I/O modules to the system:

Data Communications Module - connects a D4-454 system to devices using the DirectNET protocol, or connects as a slave to a MODBUS network.

D4-454 Communications Ports - the D4-454 CPU has 4 built-in communication ports. It allows two network connections directly from the CPU. See Chapter 3, CPU Specifications and Operation, for individual port specifications, and the sections at the end of this chapter for network connections.

MODBUS Master Module - You can use MODBUS master modules in any slot of a D4-454 system for connecting it as a master to a MODBUS network, using the RTU protocol.

MODBUS Slave Module - You can use MODBUS slave modules in any slot of a D4-454 system for connecting it as a slave to a MODBUS network, using the RTU protocol.



Module Placement and Configuration

Valid Module/Unit Locations

The most commonly used I/O modules for the D4-454 system (AC, DC, AC/DC, Relay and Analog) can be used in any base in your system. The table below lists by category the valid locations for all modules/units in a D4-454 system. Remember that the power budget can limit the number of modules in a base (discussed later).

Module Placement and Configuration

Module/Unit	Local CPU Base	Local Expansion Base	Remote I/O Base
CPUs	CPU Slot Only	–	–
Input Modules	✓	✓	✓
64pt DC Input Modules	✓ Note 1	✓ Note 1, 2	
Output Modules	✓	✓	✓
Relay Output Modules	✓	✓	✓
64pt DC Output Modules	✓ Note 1	✓ Note 1, 2	
Analog Modules	✓	✓	✓
Local Expansion			
D4-EX	–	✓	–
D4-EXDC	–	✓	–
Communications and Networking			
D4-DCM	✓ Note 3	–	–
H4-ECOM100	✓	–	–
F4-MAS-MB	✓	–	–
Remote I/O			
H4-ERM100	✓	–	–
H4-EBC	–	–	✓
D4-RM	✓	–	–
D4-RS	–	–	✓
D4-RSDC	–	–	✓
Motion			
D4-HSC	✓	✓	–
H4-CTRIO	✓	✓	✓
Specialty Modules			
D4-16SIM	✓	✓	✓
F4-16PID	✓	–	–
F4-8MPI	✓	–	–
F4-4LTC	✓	–	–
F4-CP128	✓	–	–
F4-CP128-T	✓	–	–
D4-FILL	✓	✓	✓

Note 1: When using 64 pt modules, you cannot use any specialty modules in slots 5, 6, and 7 in the same base.

Note 2: Specialty modules are allowed in expansion bases only if you are using the DL450 CPU and all bases in the system are the D4--xxB--1 type bases.

Note 3: When used with H4-ERM100 Ethernet Remote I/O system.

I/O Configuration Methods

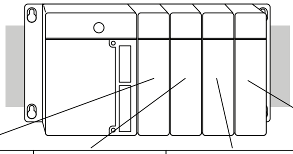
There are two methods of I/O configuration for the D4-454 CPUs.

- **Auto configuration** - the CPU automatically configures the I/O. It assigns the lowest I/O numbers to the module in slot 0 (the slot next to the CPU), the next set of I/O numbers to the next module in the base, etc. The numbers are assigned only to modules actually in the base, not to empty slots in the base. This is the default mode of the CPU.
- **Manual configuration** - The CPU allows you to assign I/O numbers. Numbers can be assigned to empty slots or in any order as long as the numbers are assigned in groups of
16 or 32.

Automatic Configuration

The D4-454 CPUs automatically detect any installed I/O modules (including specialty modules) at power up, and establish the correct I/O configuration and addresses. 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, 16, 32, or 64 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.



Slot 0 8pt. Input X0–X7	Slot 1 32pt. Output Y0–Y37	Slot 2 16pt. Input X10–X27	Slot 3 8pt. Input X30–X37

The DirectSOFT programming software allows you to view the automatic I/O configuration by using the PLC Configure I/O menu option.

Manual Configuration

It may never become necessary, but the D4-454 CPUs allow manual I/O address assignment for any I/O slot(s) in local or 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 X10 and X200.

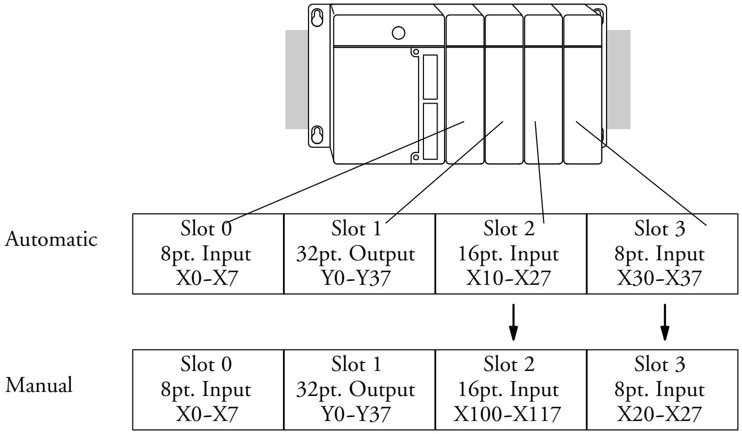
In automatic configuration, the addresses are assigned on 8-point boundaries. Manual configuration, however, assumes that all modules are at least 16 points, so you can only assign addresses that are a multiple of 20 (octal). For example, X30 and Y50 are not valid addresses. You can still use 8-point modules, but 16 addresses will be assigned and the upper 8 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 D4-454 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 simply performing an automatic configuration.

The following diagram shows how I/O addresses change after manually configuring a slot.



Automatic	Slot 0 8pt. Input X0-X7	Slot 1 32pt. Output Y0-Y37	Slot 2 16pt. Input X10-X27	Slot 3 8pt. Input X30-X37
Manual	Slot 0 8pt. Input X0-X7	Slot 1 32pt. Output Y0-Y37	Slot 2 16pt. Input X100-X117	Slot 3 8pt. Input X20-X27

Power-On I/O Configuration Check

The D4-454 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 configuration check will detect the change and print a message on the DirectSOFT screen.

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.

WARNING: You should 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.

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.

WARNING: Verify 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 the equipment.

Calculating the Power Budget

Managing your Power Resource

As you have seen, the I/O configuration depends on your choice of I/O modules, bases, and I/O location. When determining the types and quantity of I/O modules you will be using in the D4-454 system it is important to remember there is limited amount of power available from the power supply to the system. We have provided a chart to help you easily see the amount of power you will have with your CPU, Expansion Unit or Remote Slave selection. The following chart will help you calculate the amount of power you need with your I/O selections. At the end of this section you will also find 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 can resolve the problem by shifting some of the modules to an expansion base which contains another power supply.

WARNING: You should 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.

CPU Power Specifications

The following chart shows the amount of current available for the two voltages supplied on the D4-454 CPU, Expansion unit or Remote Slave unit. Use these current values 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 devices or DL405 modules that require 24VDC.

D4-454 Power Specifications		
CPUs	5V Current Supplied in mA	Auxiliary 24V Power Source Current Supplied in mA
D4-454	3100	400
D4-454DC-1	3100	None

DL405 Power Specifications		
Remote and Expansion Units	5V Current Supplied in mA	Auxiliary 24V Power Source Current Supplied in mA
D4-EX	4000	400
D4-EXDC	4000	None
D4-RS	3700	400
D4-RSDC	3700	None
H4-EBC	3470	400
H4-EBC-F	3300	400

Module Power Requirements

The chart on the next page shows the amount of maximum current required for each of the DL405 modules. Use these currents when calculating the power budget for your system. If external 24VDC is required, the external 24V from the CPU power supply may be used as long as the power budget is not exceeded.

Device	5V Current Required (mA)	External 24V Current Required (mA)
I/O Bases		
D4-04B-1	80	None
D4-06B-1	80	None
D4-08B-1	80	None
DC Input Modules		
D4-08ND3S	100	None
D4-16ND2	150	None
D4-16ND2F	150	None
D4-32ND3-1	150	None
D4-64ND2	300 (max)	None
AC Input Modules		
D4-08NA	100	None
D4-16NA	150	None
D4-16NA-1	150	None
AC/DC Input Modules		
D4-16NE3	150	None
F4-08NE3S	90	None
DC Output Modules		
F4-08TD1S	295	None
D4-16TD1	200	125
D4-16TD2	400	None
D4-32TD1	250	140
D4-42TD1-1	250	140 (5-15VDC)
D4-32TD2	350	120 / (4A max including loads)
D4-64TD1	800 (max)	None
AC Output Modules		
D4-08TA	250	None
D4-16TA	450	None
Relay Output Modules		
D4-08TR	550	None
F4-08TRS-1	575	None
F4-08TRS-2	575	None
D4-16TR	1000	None
Programming		
*DV-1000**	150	None

* Retired.

Device	5V Current Required (mA)	External 24V Current Required (mA)
Analog Modules		
F4-04AD	85	100
F4-04ADS	270	120
F4-08AD	75	90
*D4-02DA**	250	300
F4-04DA	120	180
F4-04DA-1	70	75 + 20 per channel
F4-04DA-2	90	75 + 20 per channel
F4-04DAS-1	60	50 per channel
F4-04DAS-2	60	60 per channel
F4-08DA-1	90	100 + 20 per channel
F4-16DA-1	90	100 + 20 per channel
F4-16DA-2	80	25 max.
F4-16AD-1	100	100
F4-16AD-2	75	100
F4-08THM-n	120	50 + 20 per channel
F4-08RTD	80	None
Remote I/O		
H4-ERM100	320	None
D4-RM	300	None
D4-RS	300	None
Communications and Networking		
D4-DCM	500	None
H4-ECOM100	300	None
F4-MAS-MB	235	None
CoProcessors ™		
F4-CP-128	305	None
F4-CP-512	235	None
F4-CP128-T	350	None
Specialty Modules		
D4-16SIM	150	None
D4-HSC	300	None
H4-CTRIO	400	None
F4-16PID	160	None
F4-8MPI	225	170
F4-4LTC	280	75

* Retired.

Power Budget Calculation Example

The following example shows how to calculate the power budget for the D4-454 system.

Base 0	Module Type	5VDC (mA)	Auxiliary Power Source 24VDC Output (mA)
CPU/ Expansion Unit/ Remote Slave Used	D4-454	3700	400
Slot 0	D4-16ND2	+ 150	+ 0
Slot 1	D4-16ND2	+ 150	+ 0
Slot 2	F4-04DA-1	+ 70	+ 155
Slot 3	D4-08ND3S	+ 100	+ 0
Slot 4	D4-08ND3S	+ 100	+ 0
Slot 5	D4-16TD2	+ 400	+ 0
Slot 6	D4-16TD2	+ 400	+ 0
Slot 7	D4-16TR	+ 1000	+ 0
Other			
Base	D4-08B-1	+ 80	+0
Maximum Power Required		2450	155
Remaining Power Available		3700–2450 = 1250	400–155 = 245

1. Using the tables at the beginning of the Power Budgeting section of this chapter, fill in the information for the CPU/Expansion Unit/Remote Slave, I/O modules, and any other devices that will use system power including devices that use the 24VDC output. Pay special attention to the current supplied by either the CPU, Expansion Unit, and Remote Slave since they do differ. Notice the Base falls into the “Other” category because it also has power requirements.

2. Add the current columns starting with Slot 0 and put the total in the row labeled “Maximum power required”.

3. Subtract the row labeled “Maximum power required” from the row labeled “CPU/Expansion Unit/Remote Slave Used”. Place the difference in the row labeled “Remaining Power Available”.

4. If “Maximum Power Required” is greater than “CPU/Expansion Unit/Remote Slave Used” in any of the three columns, the power budget will be exceeded. It will be unsafe to use this configuration and you will need to restructure your I/O configuration.

Power Budget Calculation Worksheet

You may copy and use the following blank chart for your power budget calculations.

Base (X)	Module Type	5VDC (mA)	Auxiliary Power Source 24VDC Output (mA)
CPU/ Expansion Unit/ Remote Slave Used			
Slot 0			
Slot 1			
Slot 2			
Slot 3			
Slot 4			
Slot 5			
Slot 6			
Slot 7			
Other			
Maximum Power Required			
Remaining Power Available			

1. Using the tables at the beginning of the Power Budgeting section of this chapter, fill in the information for the CPU/Expansion Unit/Remote Slave, I/O modules, and any other devices that will use system power including devices that use the 24VDC output. Pay special attention to the current supplied by either the CPU, Expansion Unit, and Remote Slave since they do differ. Notice the Base falls into the "Other" category because it also has power requirements.

2. Add the current columns starting with Slot 0 and put the total in the row labeled "Maximum power required".

3. Subtract the row labeled "Maximum power required" from the row labeled "CPU/Expansion Unit/Remote Slave Used". Place the difference in the row labeled "Remaining Power Available".

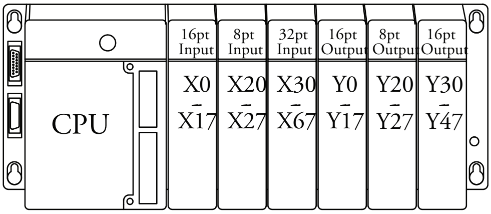
4. If "Maximum Power Required" is greater than "CPU/Expansion Unit/Remote Slave Used" in any of the three columns, the power budget will be exceeded. It will be unsafe to use this configuration and you will need to restructure your I/O configuration.

Local I/O Expansion

The following I/O base configurations will assist you in understanding the options available in the D4-454 series. Local and expanded bases are the most common and cost effective way of installing I/O. With local and expanded I/O the CPU can automatically configure the I/O for you. Use Remote I/O when it is necessary to locate I/O at distances away from the CPU. Remote I/O will require additional ladder programming to operate.

Local Base and I/O

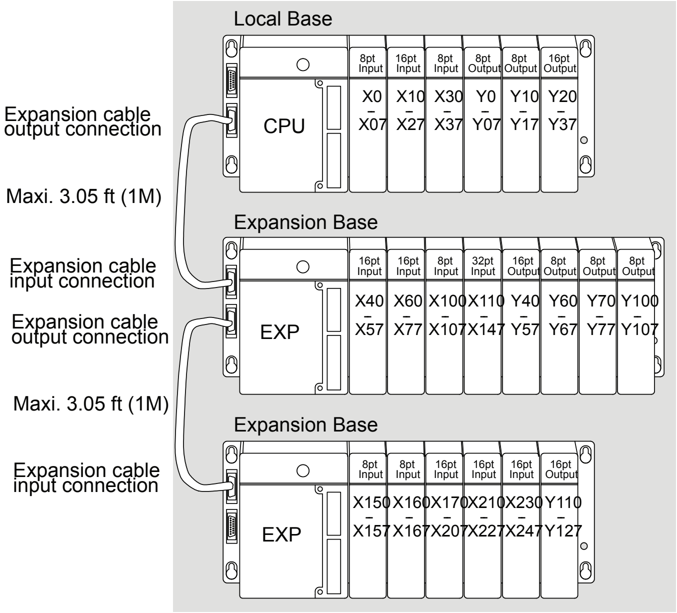
The local base is the base in which the CPU resides. Local I/O modules reside in the same base as the CPU. For Example, placing 32-point modules in all eight slots in an 8-slot will use 256 I/O points. The status of each I/O point is updated each I/O scan of the CPU.



Local Expansion Base and I/O

Use local expansion when you need more I/O points or a greater power budget than the local base provides. The expansion bases require a Local Expansion Unit (in the place of a CPU), and a cable (either D4-EXCBL-1 or D4-EXCBL-2) to connect to the local CPU base. The CPU base is always the first base in the expansion chain. The following figure shows one CPU base, two expansion bases and examples of I/O numbering.

D4-454 supports a maximum of 3 expansion bases, and maximum of 1024 input points and 1024 output points (includes local base I/O)



Remote I/O Expansion

How to Add Remote I/O Channels

Remote I/O is useful for a system that has a sufficient number of sensors and other field devices located a relative long distance away (up to 1000 meters, or 3050 feet) from the more central location of the CPU. The methods of adding remote I/O are:

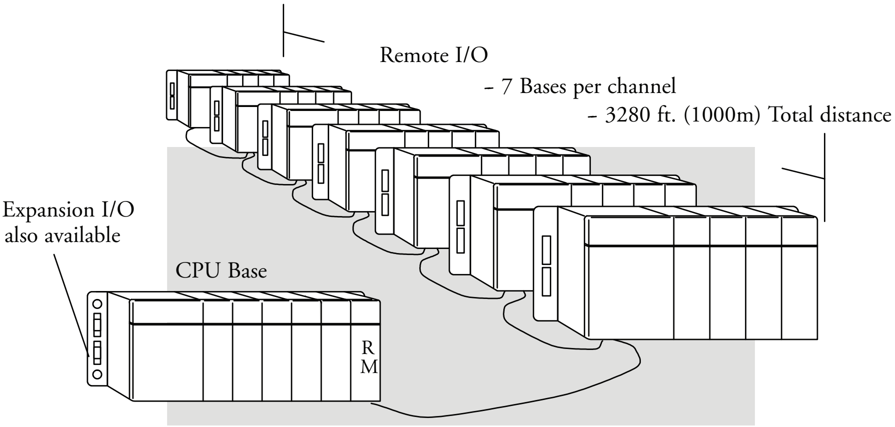
- The CPUs comm port 3 features a built-in Remote I/O channel.
- You may also use one or two D4-RM Remote Master modules in the local base.

Remote I/O	D4-454
Maximum number of Remote Masters supported in the local CPU base (1 channel per Remote Master)	2
CPU built-in Remote I/O channels	1
Maximum I/O points supported by each channel	512
Maximum Remote I/O points supported	1536
Maximum number of Remote I/O bases per channel	7

The use of Remote I/O does not limit the use of local expansion I/O discussed in the previous section. In fact, Remote I/O point numbering is assignable. Depending on the CPU scan time, remote I/O updates may be slower than local and expansion I/O, due to the serial communications involved.

Remote I/O points map into different CPU memory locations other than local / local expansion I/O. So, the addition of remote I/O does not reduce the number of local I/O points. Refer to the DL405 Remote I/O manual for details on remote I/O configuration and numbering.

The following figure shows 1 CPU base, and one remote I/O channel (D4-RM) with seven remote bases. If we had used the CPUs built-in remote I/O channel on port 3, we would not be required to add a remote master (D4-RM) module.



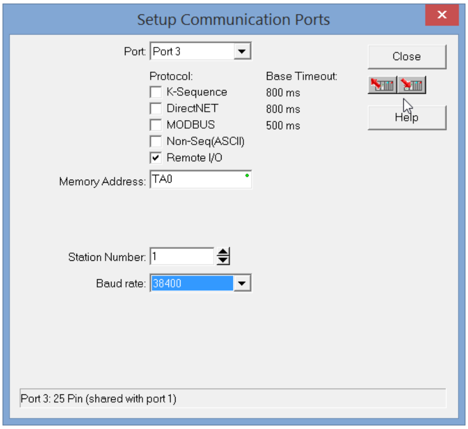
Configuring the CPU’s Remote I/O Channel

This section describes how to configure the D4-454 built-in remote I/O channel. Additional information is in the Remote I/O manual, D4-REMIO-M, which you will need in configuring the Remote slave units on the network. You can use the D4-REMIO-M exclusively when using regular Remote Master and Remote Slave modules for remote I/O in the DL405 system.

The D4-454 CPU’s built-in remote I/O channel has the same capability as a Remote Master module, the D4-RM. Specifically, it can communicate with up to seven remote bases containing a maximum of 512 I/O points, at a maximum distance of 1000 meters. If required, you can still use Remote Master modules in the local CPU base (512 I/O points on each channel), for a total of three channels providing 1536 total remote I/O points. First, we’ll need to set up the Remote I/O communications.

You may recall from the CPU specifications in Chapter 3 that the D4-454’s port 3 is capable of several protocols. To configure the port in DirectSOFT, choose the PLC menu, then Setup>Setup Secondary Comm Port.

- **Port**: From the port number list box at the top, choose “Port 3”.
- **Protocol**: Click the box to the left of “Remote I/O” to select. The dialog shown below will appear.



Setup Communication Ports

Port: Port 3

Protocol:
□ K-Sequence
□ DirectNET
□ MODBUS
□ Non-Seq(ASCII)
☑ Remote I/O

Base Timeout:
800 ms
800 ms
500 ms

Memory Address: TA0

Station Number: 1

Baud rate: 38400

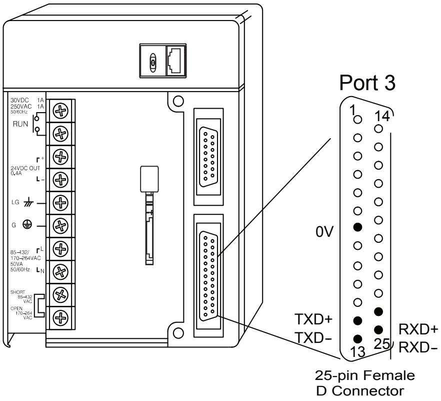
Port 3: 25 Pin (shared with port 1)

- **Station Number:** Choose “0” as the station number, which makes the D4-454 the master. Station numbers 1–7 are reserved for remote slaves.
- **Baud Rate:** The baud rates 19200 and 38400 baud 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. Important: You must configure the baud rate on the Remote Slaves (via DIP switches) to match the baud rate selection for the CPU’s port 3.
- **Memory Address**: Choose a V-memory address to use as the starting location of 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.
- Then click the button indicated to send the Port 3 configuration to the CPU, and click **Close**.

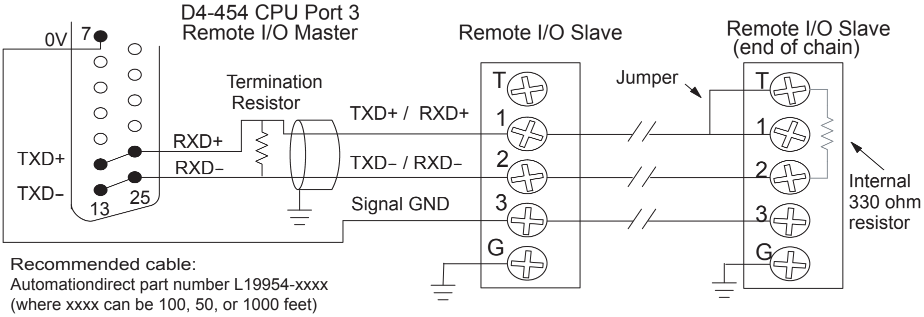
The next step is to make the connections between all devices on the Remote I/O link.

The location of Port 3 on the D4-454 is on the 25-pin connector, as pictured to the right. Remember that ports 1 and 3 are "logical" ports that share the 25-pin connector. Port 3 is an RS-422 non-isolated port. The pin assignments are:

- Pin 7 Signal GND
- Pin 12 TXD+
- Pin 13 TXD-
- Pin 24 RXD+
- Pin 25 RXD-



Now we are ready to discuss wiring the D4-454 to the remote slaves on the remote base(s). The remote I/O link is a 3-wire, half duplex type. Since Port 3 of the D4-454 CPU is a 5-wire full duplex-capable port, we must jumper its transmit and receive lines together as shown below (converting it to 3-wire half-duplex).



Recommended cable:
Automationdirect part number L19954-xxxx
(where xxxx can be 100, 50, or 1000 feet)
Belden 9842 or equivalent

The twisted/shielded pair connects to the D4-454 Port 3 as shown. Be sure to connect the cable shield wire to the signal ground connection. 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. Be sure to add the jumper at the last slave to connect the required internal termination resistor

For more details and information on Remote I/O link wiring, please refer to the Remote I/O link user manual D4-REMIO-M available for free download from:
www.automationdirect.com.

Configure Remote I/O Slaves

After configuring the D4-454 CPUs Port 3 and wiring it to the remote slave(s), use the following checklist to complete the configuration of the remote slaves. Full instructions for these steps are in the Remote I/O manual.

Set the baud rate DIP switches to match CPUs Port 3 setting.

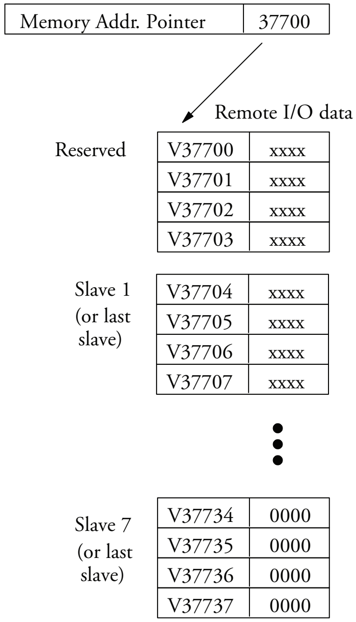
Select a station address for each slave, from 1 to 7. Each device on the remote link must have a unique station address. There can be only one master (address 0) on the remote link.

If you're familiar with configuring remote bases, then you'll recall the fixed table location in V-memory (V7404 - V7477) to configure up to two remote I/O channels. However, we use a separate table for configuring the D4-454 CPUs built-in-remote I/O channel. You will still need the table at V7404 to configure any Remote Master modules.

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 3 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.



Memory Addr. Pointer	37700

Remote I/O data

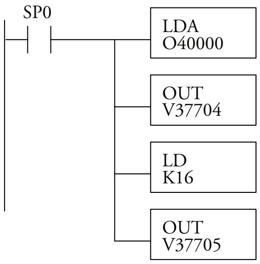
Reserved	V37700	xxxx
	V37701	xxxx
	V37702	xxxx
	V37703	xxxx
Slave 1 (or last slave)	V37704	xxxx
	V37705	xxxx
	V37706	xxxx
	V37707	xxxx
Slave 7 (or last slave)	V37734	0000
	V37735	0000
	V37736	0000
	V37737	0000

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

- Starting address of slave's input data
- Number of slave's input points
- Starting address of outputs in slave
- Number of slave's 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 3-slave system will have a remote configuration table containing 4 reserved words, 12 words of data and 16 words of "0000".

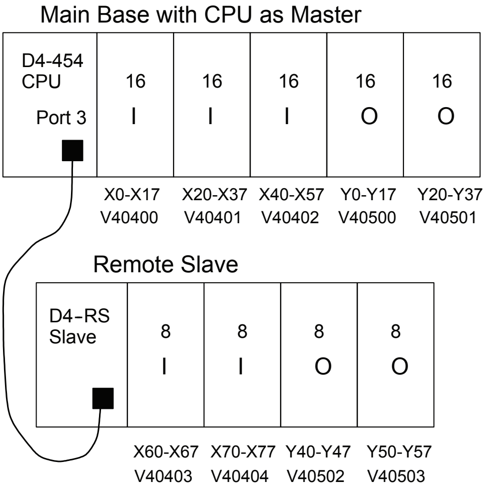
A portion of the ladder program must configure this table (just once) at power up. Use the LDA instruction as shown to the right, to load an address to place in the table. Use the regular LD constant to load the number of slave's input or output points.

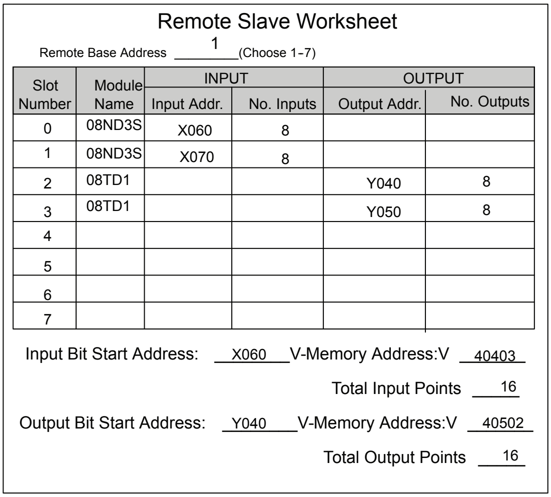


The D4-REMIO-M manual contains thorough examples for configuring the table at V7404, which you can adapt for this table as well. The following page gives a shorter program example for one slave.

Consider the simple system featuring the Remote I/O shown below. The D4-454 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 38400kB.

We can map the remote I/O points as any type of I/O point, simply by choosing the appropriate range of V-memory. Remember that on the D4-454, you have both GX and GY data types available. 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 Input Addr.	No. Inputs	OUTPUT Output Addr.	No. Outputs
0	08ND3S	X060	8		
1	08ND3S	X070	8		
2	08TD1			Y040	8
3	08TD1			Y050	8
4					
5					
6					
7					

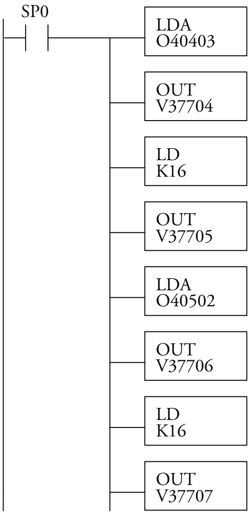
Input Bit Start Address: X060 V-Memory Address:V 40403

Total Input Points 16

Output Bit Start Address: Y040 V-Memory Address:V 40502

Total Output Points 16

Remote I/O Setup Program

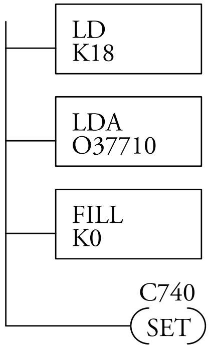


Using the Remote Slave worksheet shown above can help organize our system data in preparation for writing our ladder program (a blank full-page copy of this worksheet is in Appendix A of the D4-REMIO-M manual for your use and duplication). The four key parameters we need to place in our Remote I/O configuration table is 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 Specification 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.

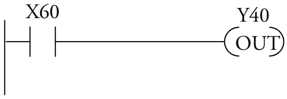


We continue our setup program from the previous page by adding a segment which fills the remainder of the table with zeros. The easiest way is to use the fill command as shown. The example to right fills zeros for slave numbers 2–7, which do not exist in our example system

(6 bases x 4 = 24 locations, = 18 hex).

On the last rung in the example program above, we set a special relay contact C740. This particular contact indicates to the CPU that the ladder program has just 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 setup program.

Remote I/O Test Program



Now we can verify the remote I/O link and setup program operation. A simple quick check can be done with just 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.

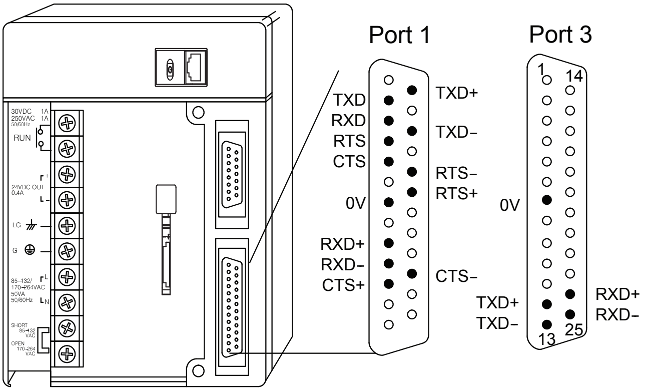
Network Connections to MODBUS® and DirectNet

Configuring the CPU's Comm Ports

This section describes how to configure the CPU's built-in networking port for either MODBUS or DirectNET. This will allow you to connect the D4-454 PLC system directly to MODBUS networks using the RTU protocol, or to other devices on a DirectNET network. MODBUS hosts systems on the network must be capable of issuing the MODBUS commands to read or write the appropriate data. For details on the MODBUS protocol, please refer to Gould MODBUS protocol reference Guide (P1-MBUS-300 Rev. J). In the event a more recent version is available, check with your MODBUS supplier before ordering the documentation. For more details on DirectNet, order our DirectNET manual, part number DA-DNET-M.

***NOTE:** For information about the MODBUS protocol, visit the Modbus.org website. The PI-MBUS-300 Rev J Modicon Modbus Protocol Reference Guide can be downloaded from: Modbus.org/doc/PI_MBUS_300.pdf. For more information about DirectNET protocol, order our DirectNET user manual, part number DA-DNET-M, or download it free from our website at: www.automationdirect.com. Select Manuals/Docs > Product Manuals > Miscellaneous >DA-DNET-M.*

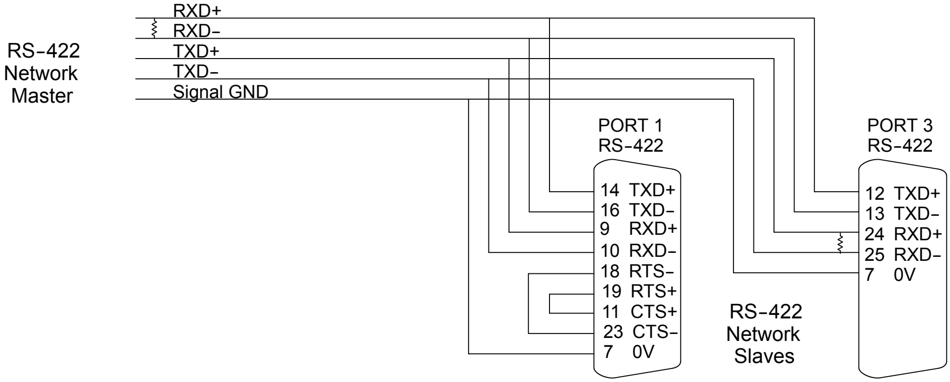
The D4-454 CPUs Port 1, Port 2 and Port 3 can operate as a master or slave for both MODBUS and DirectNET. Port 1 has RS-232 and RS-422 signal levels available on separate pins, Port 2 uses RS-232 signal levels, and Port 3 uses RS422 signal levels. Ports 1 and Port 3 on the D4-454 share the 25-pin D-shell connector, as shown below. You can not simultaneously use Port 1's RS-232 signals and its RS-422 signals.

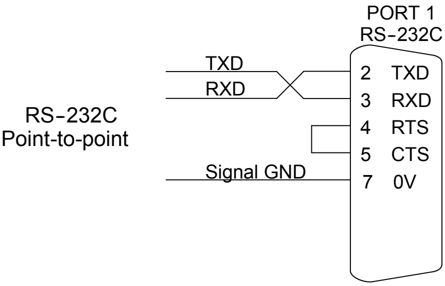


***NOTE:** The recommended cable for RS-232 or RS-422 is AutomationDirect part number L19772-xxxx, where xxxx can be 100, 500 or 1000 feet (equivalent to Belden 8102).*

You will need to determine whether the network connection is a 3-wire RS-232 type, or a 5-wire RS-422 type. Normally, we use RS-232 signals for shorter distances (15 meters max.), for communications between just two devices. Use RS-422 signals for longer distances (1000 meters max.) and for multi-drop networks (from 2 to 248 devices). Be sure to use termination resistors at the both ends of RS-422 network wiring, matching the impedance rating of the cable (between 100 and 500 ohms).

NOTE: *If the D4-454 is to be used as a MODBUS Master and the distance will be more than 1,000 feet, you can use the MODBUS Network Master module, F4-MAS-MB, and use the RS-485 port. See the module on our website, automationdirect.com for more details.*

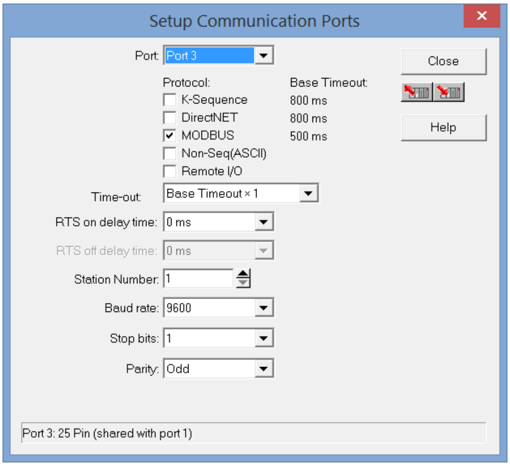




MODBUS Port Configuration

In DirectSOFT, choose the PLC menu, then Setup > Secondary Comm Port.

- **Port:** From the port number list box, choose Port 1, 2 or 3.
- **Protocol:** Click the box to the left of Modbus to select it. The dialog below will appear.



Setup Communication Ports

Port: Port 3

Protocol:	Base Timeout:
☐ K-Sequence	800 ms
☐ DirectNET	800 ms
☑ MODBUS	500 ms
☐ Non-Seq(ASCII)	
☐ Remote I/O	

Time-out: Base Timeout × 1

RTS on delay time: 0 ms

RTS off delay time: 0 ms

Station Number: 1

Baud rate: 9600

Stop bits: 1

Parity: Odd

Port 3: 25 Pin (shared with port 1)

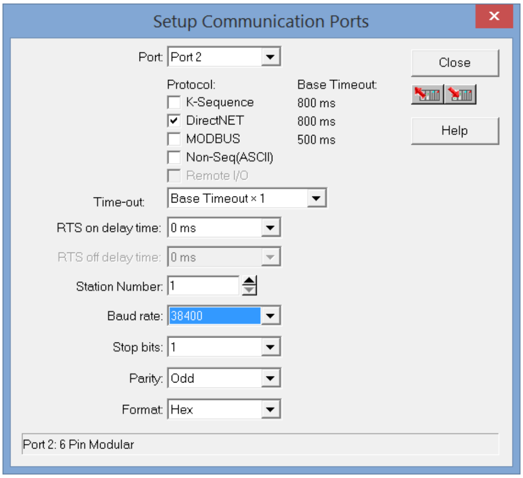
- **Timeout:** the amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS ON / OFF Delay Time:** The RTS ON Delay Time specifies the time the D4-454 waits to send the data after it has raised the RTS signal line. The RTS OFF Delay Time specifies the time the D4-454 waits to release the RTS signal line after the data has been sent.
- **Station Number:** For making the CPU port a MODBUS master, choose “1”. The possible range for MODBUS slave numbers is from 1 to 90. Each slave must have a unique number. At power up, the port is automatically a slave, unless and until the D4-454 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 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.

Then click the “write to PLC” button to send the Port configuration to the CPU, and click Close.

DirectNET Port Configuration

In DirectSOFT, choose the PLC menu, then Setup > Secondary Comm Port.

- **Port:** From the port number list box, choose Port 1, 2 or 3.
- **Protocol:** Click the box to the left of DirectNET to select it. The dialog below will appear.



Setup Communication Ports

Port: Port 2

Protocol: □ K-Sequence ☑ DirectNET □ MODBUS □ Non-Seq(ASCII) □ Remote I/O

Base Timeout: 800 ms, 800 ms, 500 ms

Time-out: Base Timeout × 1

RTS on delay time: 0 ms

RTS off delay time: 0 ms

Station Number: 1

Baud rate: 38400

Stop bits: 1

Parity: Odd

Format: Hex

Port 2: 6 Pin Modular

- **Timeout:** the amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS ON / OFF Delay Time:** The RTS ON Delay Time specifies the time the D4-454 waits to send the data after it has raised the RTS signal line. The RTS OFF Delay Time specifies the time the D4-454 waits to release the RTS signal line after the data has been sent.
- **Station Number:** For making the CPU port a DirectNET master, choose “1”. The allowable range for DirectNET slaves is from 1 to 90 (each slave must have a unique number). At power up, the port is automatically a slave, unless and until the D4-454 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 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 between hex or ASCII formats.

Then click the “write to PLC” button to send the Port configuration to the CPU, and click Close.

Network Slave Operation

This section describes how other devices on a network can communicate with a CPU port that you have configured as a DirectNET slave or MODBUS slave. A MODBUS host must use the MODBUS RTU protocol to communicate with the D4-454 as a slave. The host software must send a MODBUS function code and MODBUS address to specify a PLC memory location the D4-454 comprehends. The DirectNET host just uses normal I/O addresses to access the D4-454. No CPU ladder logic is required to support either MODBUS slave or DirectNET slave operation.

NOTE: *For more information on DirectNET proprietary protocol, see the DirectNET reference manual, DA-DNET-M, available on our website.*

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 D4-454 supports the MODBUS function codes described below.

MODBUS Function Code	Function	D4-454 Data Types Available
01	Read a group of coils	Y, CR, T, CT, GY
02	Read a group of inputs	X, SP, GX
05 (slave only)	Set / Reset a single coil	Y, CR, T, CT
15	Set / Reset a group of coils	Y, CR, T, CT
03, 04	Read a value from one or more registers	V
06 (slave only)	Write a value into a single register	V
16	Write a value into a group of registers	V

MODBUS Data Types Supported

The memory types in a D4-454 system include X input, Y output, C control relay, V-memory data registers, etc. MODBUS uses differently named data types. So, you will need to determine which MODBUS data types corresponds to any desired PLC memory location by using the cross-reference table below.

D4-454 Memory Type	Quantity (Decimal)	PLC Range (Octal)	Corresponding MODBUS Data Type	RX Function Code
Inputs (X)	1024	X0 – X1777	Input	02
Global Inputs (GX)	1536	GX0 – GX2777	Input	02
Special Relays (SP)	512	SP0 – SP137 SP320 – SP717	Input	02
Outputs (Y)	1024	Y0 – Y1777	Coil	01
Global Outputs (GY) *	1536	GY0 – GY2777	Coil	01
Control Relays (CR)	2048	C0 – C3777	Coil	01
Timer Contacts (T)	256	T0 – T377	Coil	01
Counter Contacts (CT)	256	CT0 – CT377	Coil	01
Stage Status Bits (S)	1024	S0 – S1777	Coil	01
Timer Current Values (V)	256	V0 – V377	Input Register	03
Counter Current Values (V)	256	V1000 – V1377	Input Register	03
V-Memory, user data (V)	3072 12288	V1400 – V7377 V10000 – V37777	Holding Register	03
V-Memory, system (V)	320	V700 – V777 V7400 – V7777	Holding Register	03

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 types 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 corresponds to the PLC memory location. This is the easiest method, but not all packages allow you to do it this way. The various MODBUS data types were presented earlier, but they have been included again in the following table.

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, CR, S, T, C (contacts)
- Word - V, Timer current value, Counter current value

In either case, you basically just convert the PLC octal address to decimal and add the appropriate MODBUS address (if required). The table on the next page shows the exact equation used for each group of data.

D4-454 Memory Type	QTY (Decimal)	PLC Range (Octal)	MODBUS Address Range (Decimal)	MODBUS Data Type
For Discrete Data Types	**Convert PLC Address to Decimal**		**+ Start of Range**	**+ Data**
Inputs (X)	1024	X0–X1777	2048–3071	Input
Special Relays (SP)	512	SP0–SP137 SP320–SP717	3072–3167 3280–3535	Input
Outputs (Y)	1024	Y0–Y1777	2048–3071	Coil
Control Relays (CR)	2048	C0–C3777	3072–5119	Coil
Timer Contacts (T)	256	T0–T377	6144–6399	Coil
Counter Contacts (CT)	256	CT0–CT377	6400–6655	Coil
Stage Status Bits (S)	1024	S0–S1777	5120–6143	Coil
Global Inputs (GX) *	1536	GX0–GX2777	0–1535	Input
Global Outputs (GY) *	1536	GY0–GY2777	0–1535	Coil
For Word Data Types ----------	**Convert PLC Address to Decimal**		**+**	**Data Type**
Timer Current Values (V)	256	V0–V377	0–255	Input Register
Counter Current Values (V)	256	V1000–V1377	512–767	Input Register
V-Memory, user data (V)	3072 12288	V1400–V7377 V10000–V37777	768–3839 4096–16383	Holding Register
V-Memory, system (V)	320	V700–V777 V7400–V7777	448–768 3480–3735	Holding Register

* The total of GX and GY global I/O points cannot exceed 1536 points.

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.

PLC Address (Dec.) = Data Type

1. Find V-memory in the table.
2. Convert V2100 into decimal (1088).
3. Use the MODBUS data type from the table.

V2100 = 1088 decimal

1088 + Hold. Reg. = **Holding Reg. 1088**

V-Memory, user date (V)	3072 12288	V4100 - V7377 V10000-V37777	768 - 3839 4096 - 16383	Holding Register

Example 2: Y20

Find the MODBUS address for output Y20

PLC Address (Dec.) + Start Addr. + Data Type

1. Find Y outputs in the table.
2. Convert Y20 into decimal (16).
3. Add the starting address for the range (2048).
4 Use the MODBUS data type from the table.

Y20 = 16 decimal

16 + 2048 = **Coil 2064**

Outputs (Y)	1024	Y0 - Y1777	2048 - 3071	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. 8

Timer Current Values (V)	256	V0 - V3777	0 - 255	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 (3072)
4. Use the MODBUS data type from the table.

PLC Address (Dec.) + Start Addr. + Data Type
V2100 = 1088 decimal
1088 + Hold. Reg. = Holding Reg. 1088

Control Relays (CR)	2048	C0 - C3777	3072 - 5119	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's still fairly simple. Basically, MODBUS also 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 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, CR, S, T (contacts), C (contacts)
- **Word** - V, Timer current Value, Counter current value

In either case, you simply convert the PLC octal address to decimal and add the appropriate MODBUS address (as required). The table on the next page shows the exact equation used for each group of data.

Discrete Data Types				
Memory Type	PLC Range (Octal)	Address (484 Mode)	Address (585/984 Mode)	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 (CR)	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
V – memory (Data Words)	V1400 – V1746	3769 / 4769	30769 / 40769
V – memory (Data Words)	V1747 – V1777	–	31000 / 41000
V – memory (Data Words)	V2000 – V7377	–	41025
V – memory (Data Words)	V10000 – V17777	–	44097
* Modbus Function 04			

NOTE: *For an automated MODBUS/KOYO address conversion utility, go to our website, www.automationdirect.com, and download the EXCEL file: modbus_conversion.xls located at: Tech Support > Technical and Application Notes > AN-MISC-010.*

Example 1: V2100 584/984 Mode

Find the MODBUS address for User V location V2100. PLC Address (Dec.) + Mode Address

1. Find V-memory in the table. V2100 = 1088 decimal
2. Convert V2100 into decimal (1088). 1088 + 40001 = $\boxed{41089}$
3. Add the MODBUS starting address for the mode (40001).

For Word Data Types....	PLC Address (Dec.)	+	Appropriate Mode Address			
Timer Current Values (V)	128	V0 – V177	0 – 127	3001	30001	Input Register
Counter Current Values (V)	128	V1200 – V7377	640 – 3839	3001	30001	Input Register
V-memory, user data (V)	1024	V2000 – V3777	1024 – 2047	4001	40001	Holding Register

Example 2: Y20 584/984 Mode

Find the MODBUS address for output Y20. PLC Addr. (Dec.) + Start Address + Mode

1. Find Y outputs in the table.
2. Convert Y20 into decimal (16). Y20 = 16 decimal
3. Add the starting address for the range (2048). 16 + 2048
4. Add the MODBUS address for the mode (1). 16 + 2048 + 1 = $\boxed{2065}$

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 PLC Address (Dec.) + Mode
Address
current value from Timer T10.

1. Find Timer Current Values in the table.
2. Convert T10 into decimal (8). T10 = 8 decimal
3. Add the MODBUS starting address for the mode (3001). 8 + 3001 = $\boxed{3009}$

For Word Data Types....	PLC Address (Dec.)	+	Appropriate Mode Address			
Timer Current Values (V)	128	V0 – V177	0 – 127	3001	30001	Input Register
Counter Current Values (V)	128	V1200 – V7377	512 – 639	3001	30001	Input Register
V-memory, user data (V)	1024	V2000 – V3777	1024 – 2047	4001	40001	Holding Register

Example 4: C54 584/984 Mode

Find the MODBUS address for Control Relay C5. PLC Addr. (Dec.) + Start Address + Mode

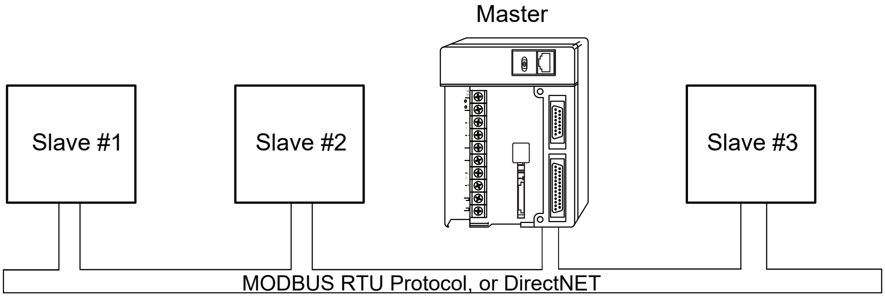
1. Find Control Relays in the table.
2. Convert C54 into decimal (44). C54 = 44 decimal
3. Add the starting address for the range (3072). 44 + 3072
4. Add the MODBUS address for the mode (1). 44 + 3072 + 1 = $\boxed{3117}$

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

Network Master Operation

This section describes how the D4-454 can communicate on a MODBUS or DirectNET 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 DirectNet 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 D4-454 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 and 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.



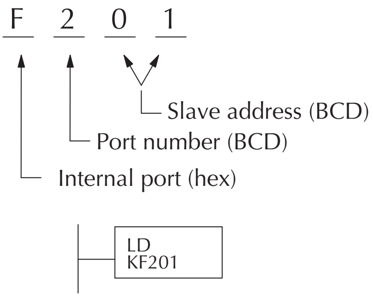
It's possible to use Ports 1, 2 and 3 for either MODBUS or DirectNET, and it is possible to use all three ports at the same time. You must tell the WX and RX instructions the intended port for each communications transaction.

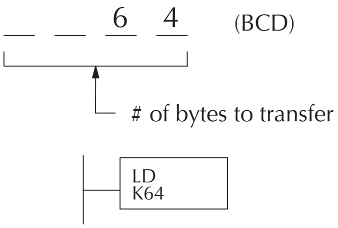
To summarize, the RLL instructions identify the following items.

1. Port number on the master (Port 1, 2 or 3), and the slave station address (LD instruction).
2. Amount of data (in bytes) you want to transfer (LD instruction).
3. Area of memory to be used by the master (LDA instruction).
4. Area of CPU V-memory to be used in communication with the slave and whether it is a write or read operation (WX or RX instruction).
5. Interlocks for communication timing for multiple WX and RX routines.

Step 1: Identify Master Port # and Slave

The first Load (LD) instruction identifies the communications port number on the network master (D4-454) and the address of the slave station. This instruction can address up to 99 MODBUS slaves, or 90 DirectNET slaves. The format of the word is shown to the right. The “F” in the upper nibble indicates the port is internal to the CPU (and not in a slot in the base). The second nimble indicates the port number, 1, 2, or 3. In this example, the “F2” in the upper byte indicates the CPU will use the RJ12 port on the CPU, 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 D4-454 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 D4-454 CPUs.

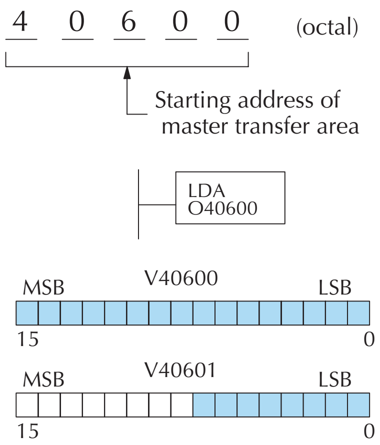
D4-454 Memory	Bits per unit	Bytes
V-memory *T / C current value*	16	2
Inputs (X, SP)	8	1
Outputs *(Y, C, Stage, T/C bits)*		
Scratch Pad Memory		
Diagnostic Status		

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.

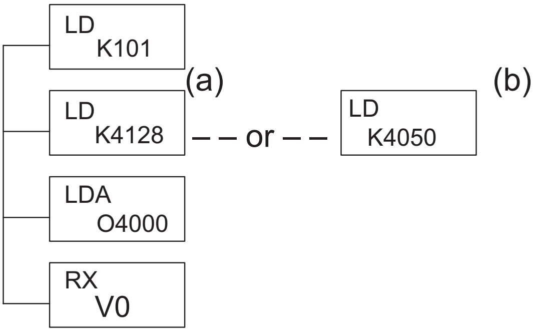
For a WX instruction, the D4-454 CPU sends the number of bytes previously specified from its memory area beginning at the LDA address specified.

For an RX instruction, the D4-454 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.*

When using MODBUS, the RX instructions uses function 3 by default, to read MODBUS holding registers (Address 40001). The DL05/06, DL250-1/260, DL350, D4-454 support function 04, read input register (Address 30001). To use function 04, put the number '4' into the most significant position (4xxx) of the total number of bytes. Four digits must be entered for the instruction to work properly with this mode.



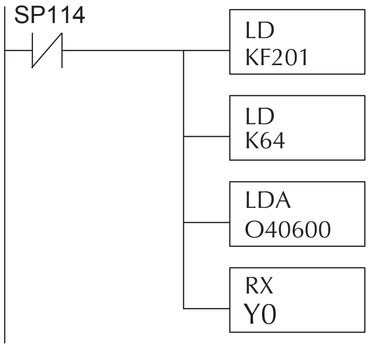
The (a) K4128 indicates the instruction will read 128 bytes of MODBUS input registers (30001). The (b) K4050 indicates the instruction will read 50 bytes of MODBUS input registers (30001). The value of 4 in the most significant position will cause the RX to use MODBUS function 4 (30001 range).

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.

The RX instruction reads data from the slave starting at the address specified. The WX instruction writes data to the slave starting at the address specified.

- ***DirectNET*** slaves – specify the same address in the WX and RX instruction as the slave's native I/O address
- MODBUS DL405, DL205, or DL06 slaves – specify the same address in the WX and RX instruction as the slave's native I/O address

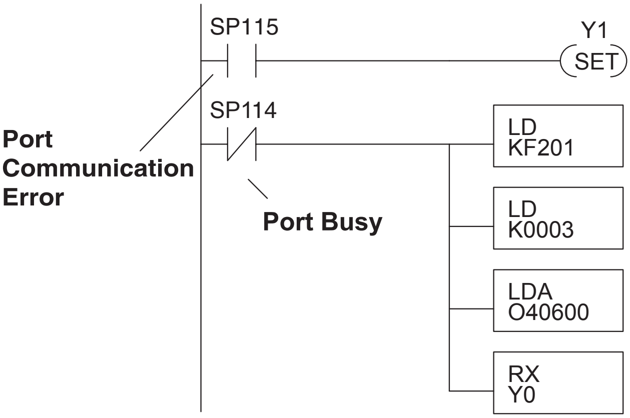


Communications from a Ladder Program

Typically network communications will last longer than 1 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 (see the Special Relays Appendix for more details). One indicates "Port busy"(SP114), and the other indicates "Port Communication Error"(SP115). The example above 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.

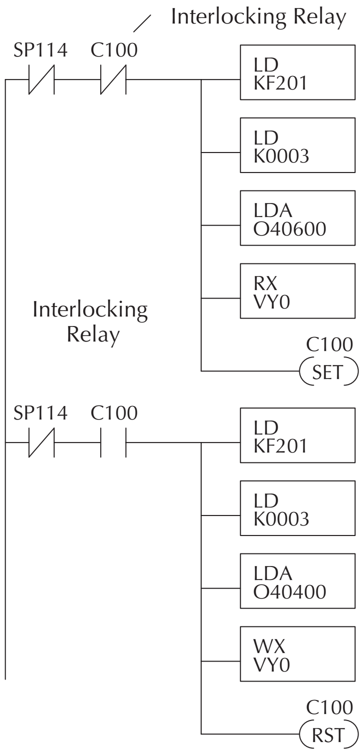


Multiple Read and Write Interlocks

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 below, 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 RLLPLUS 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.



STANDARD RLL INSTRUCTIONS

CHAPTER 5

In This Chapter...

Introduction

D4-454 PLCs offer a wide variety of instructions to perform many different types of operations. This chapter shows you how to use each standard Relay Ladder Logic (RLL) instruction. In addition to these instructions, you may also need to refer to the Drum instruction in Chapter 6, the Stage programming instructions in Chapter 7, PID in Chapter 10 and programming for analog modules in D4-ANLG-M.

There are two ways to quickly find the instruction you need.

- If you know the instruction category (Boolean, Comparative Boolean, etc.), just use the title 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(s) that discusses the instruction.

Instruction	Page	Instruction	Page
Accumulating Fast Timer (TMRAF)	5-41	And Positive Differential (ANDPD)	5-22
Accumulating Timer (TMRA)	5-41	And Store (ANDSTR)	5-16
Add (ADD)	5-86	And with Stack (ANDS)	5-72
Add Binary (ADDB)	5-99	Arc Cosine Real (ACOSR)	5-119
Add Binary Double (ADDBD)	5-100	Arc Sine Real (ASINR)	5-118
Add Binary Top of Stack (ADDBS)	5-114	Arc Tangent Real (ATANR)	5-119
Add Double (ADDD)	5-87	Binary (BIN)	5-127
Add Formatted (ADDF)	5-106	Binary Coded Decimal (BCD)	5-128
Add Real (ADDR)	5-88	Binary to Real Conversion (BTOR)	5-131
Add to Top (ATT)	5-164	Break (BREAK)	5-176
Add Top of Stack (ADDS)	5-110	Compare (CMP)	5-81
And (AND)	5-14	Compare Double (CMPD)	5-82
And Bit-of-Word (AND)	5-15	Compare Formatted (CMPF)	5-83
And (AND)	5-30	Compare Real Number (CMPR)	5-85
And (AND logical)	5-69	Compare with Stack (CMPS)	5-84
And Double (ANDD)	5-70	Cosine Real (COSR)	5-118
And Formatted (ANDF)	5-71	Counter (CNT)	5-44
And If Equal (ANDE)	5-27	Data Label (DLBL)	5-189
And If Not Equal (ANDNE)	5-27	Date (DATE)	5-173
And Immediate (ANDI)	5-32	Decode (DECO)	5-126
AND Move (ANDMOV)	5-169	Decrement (DEC)	5-98
And Negative Differential (ANDND)	5-22	Decrement Binary (DECB)	5-105
And Not (ANDN)	5-14	Degree Real Conversion (DEGR)	5-133
And Not Bit-of-Word (ANDN)	5-15	Disable Interrupts (DISI)	5-186
And Not (ANDN)	5-30	Divide (DIV)	5-95
And Not Immediate (ANDNI)	5-32	Divide Binary (DIVB)	5-104

Instruction	Page	Instruction	Page
Divide Binary by Top OF Stack (DIVBS)	5–117	Load Label (LDLBL)	5–142
Divide by Top of Stack (DIVS)	5–113	Load Real Number (LDR)	5–63
Divide Double (DIVD)	5–96	Master Line Reset (MLR)	5–183
Divide Formatted (DIVF)	5–109	Master Line Set (MLS)	5–183
Divide Real (DIVR)	5–97	Move Block (MOVBLK)	5–191
Enable Interrupts (ENI)	5–185	Move (MOV)	5–141
Encode (ENCO)	5–125	Move Memory Cartridge (MOVMC)	5–142
End (END)	5–175	Multiply (MUL)	5–92
Exclusive Or (XOR)	5–77	Multiply Binary (MULB)	5–103
Exclusive Or Double (XORD)	5–78	Multiply Binary Top of Stack (MULBS)	5–116
Exclusive Or Formatted (XORF)	5–79	Multiply Double (MULD)	5–93
Exclusive OR Move (XORMOV)	5–169	Multiply Formatted (MULF)	5–108
Exclusive Or with Stack (XORS)	5–80	Multiply Real (MULR)	5–94
Fault (FAULT)	5–188	Multiply Top of Stack (MULS)	5–112
Fill (FILL)	5–148	No Operation (NOP)	5–175
Find (FIND)	5–149	Not (NOT)	5–19
Find Block (FINDB)	5–171	Numerical Constant (NCON)	5–189
Find Greater Than (FDGT)	5–150	Or (OR)	5–12
For / Next (FOR) (NEXT)	5–178	(OR)	5–29
Goto Label (GOTO) (LBL)	5–177	Or (OR logical)	5–73
Goto Subroutine (GTS) (SBR)	5–180	Or Bit-of-Word (OR)	5–13
Gray Code (GRAY)	5–138	Or Double (ORD)	5–74
HEX to ASCII (HTA)	5–135	Or Formatted (ORF)	5–75
Increment (INC)	5–98	Or If Equal (ORE)	5–26
Increment Binary (INCB)	5–105	Or If Not Equal (ORNE)	5–26
Interrupt (INT)	5–185	Or Immediate (ORI)	5–31
Interrupt Return (IRT)	5–185	OR Move (ORMOV)	5–169
Interrupt Return Conditional (IRTC)	5–185	Or Negative Differential (ORND)	5–21
Invert (INV)	5–129	Or Not (ORN)	5–12
Load (LD)	5–57	Or Not (ORN)	5–29
Load Accumulator Indexed (LDX)	5–61	Or Not Bit-of-Word (ORN)	5–13
Load Accumulator Indexed from Data Constants (LDSX)	5–62	Or Not Immediate (ORNI)	5–31
Load Address (LDA)	5–60	Or Out (OROUT)	5–17
Load Double (LDD)	5–58	Or Out Immediate (OROUTI)	5–33
Load Formatted (LDF)	5–59	Or Positive Differential (ORPD)	5–21
Load Immediate (LDI)	5–36	Or Store (ORSTR)	5–16
Load Immediate Formatted (LDIF)	5–37	Or with Stack (ORS)	5–76

Instruction	Page	Instruction	Page
Out (OUT)	5–17	Source to Table (STT)	5–158
Out Bit-of-Word (OUT)	5–18	Square Root Real (SQRTR)	5–119
Out (OUT)	5–64	Stage Counter (SGCNT)	5–46
Out Double (OUTD)	5–64	Stop (STOP)	5–175
Out Formatted (OUTF)	5–65	Store (STR)	5–10, 5-28
Out Immediate (OUTI)	5–33	Store Bit-of-Word (STRB)	5–11
Out Immediate Formatted (OUTIF)	5–34	Store If Equal (STRE)	5–25
Out Indexed (OUTX)	5–66	Store If Not Equal (STRNE)	5–25
Out Least (OUTL)	5–67	Store Immediate (STRI)	5–31
Out Most (OUTM)	5–67	Store Negative Differential (STRND)	5–20
Pause (PAUSE)	5–19	Store Not (STRN)	5–28
Pop (POP)	5–68	Store Not (STRN)	5–10
Positive Differential (PD)	5–19	Store Not Bit-of-Word (STRNB)	5–11
Print Message (PRINT)	5–192	Store Not Immediate (STRNI)	5–31
Radian Real Conversion (RADR)	5–133	Store Positive Differential (STRPD)	5–20
Read from Intelligent Module (RD)	5–196	Subroutine Return (RT)	5–180
Read from Network (RX)	5–198	Subroutine Return Conditional (RTC)	5–180
Real to Binary Conversion (RTOB)	5–132	Subtract (SUB)	5–89
Remove from Bottom (RFB)	5–155	Subtract Binary (SUBB)	5–101
Remove from Table (RFT)	5–161	Subtract Binary Double (SUBBD)	5–102
Reset (RST)	5–23	Subtract Binary Top of Stack (SUBBS)	5–115
Reset Bit-of-Word (RST)	5–24	Subtract Double (SUBD)	5–90
Reset Immediate (RSTI)	5–35	Subtract Formatted (SUBF)	5–107
Reset Watch Dog Timer (RSTWT)	5–176	Subtract Real (SUBR)	5–91
Rotate Left (ROTL)	5–123	Subtract Top of Stack (SUBS)	5–111
Rotate Right (ROTR)	5–124	Sum (SUM)	5–120
Reset Bit (RSTBIT)	5–144	Swap (SWAP)	5–172
Segment (SEG)	5–137	Table Shift Left (TSHFL)	5–167
Set (SET)	5–23	Table Shift Right (TSHFR)	5–167
Set Bit-of-Word (SET)	5–24	Table to Destination (TTD)	5–152
Set Immediate (SETI)	5–35	Tangent Real (TANR)	5–118
Set Bit (SETBIT)	5–146	Ten’s Complement (BCDCPL)	5–130
Shift Left (SHFL)	5–121	Time (TIME)	5–174
Shift Register (SR)	5–50	Timer (TMR) and Timer Fast (TMRF)	5–40
Shift Right (SHFR)	5–122	Up Down Counter (UDC)	5–48
Shuffle Digits (SFLDGT)	5–139	Write to Intelligent I/O Module (WT)	5–197
Sine Real (SINR)	5–118	Write to Network (WX)	5–200

Using Boolean Instructions

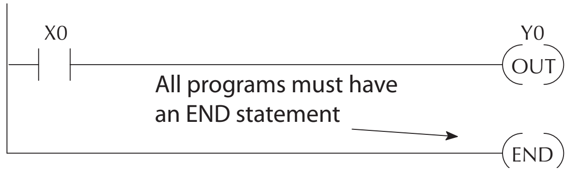
Do you ever wonder why so many PLC manufacturers always quote the scan time for a 1K Boolean program? Simple. Most programs utilize many Boolean instructions. These are typically very simple instructions designed to join input and output contacts in various series and parallel combinations. Our DirectSOFT software 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 DirectSOFT, but are implied. In other words, they are not actually keyboard commands, however, they can be seen in a Mnemonic View of the program once the DirectSOFT program has been developed and accepted (compiled).

The following paragraphs show how these instructions are used to build simple ladder programs.

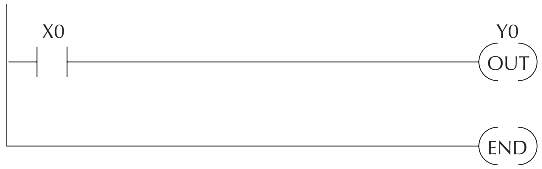
END Statement

D4-454 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. This chapter will discuss the instruction set in detail.



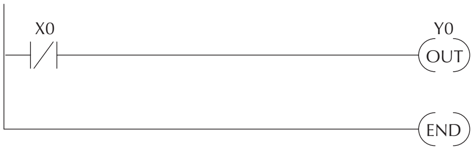
Simple Rungs

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.



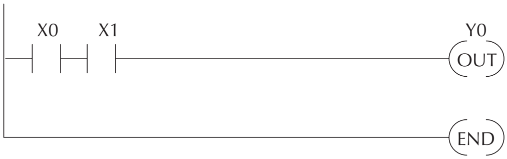
Normally Closed Contact

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.



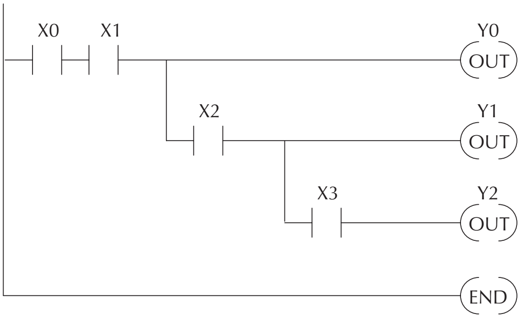
Contacts in Series

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.



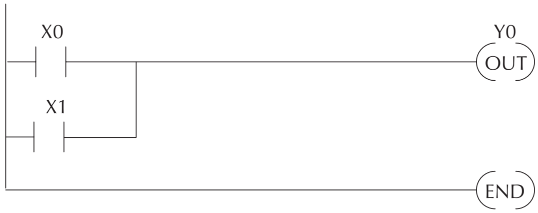
Midline Outputs

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.



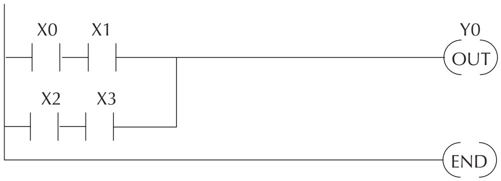
Parallel Elements

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.



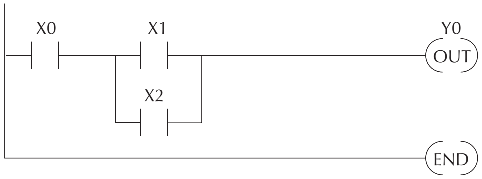
Joining Series Branches in Parallel

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.



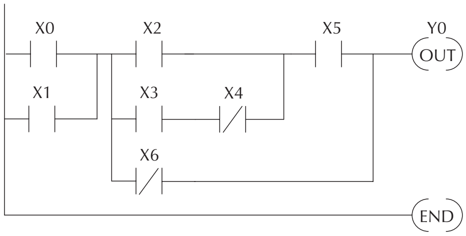
Joining Parallel Branches in Series

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

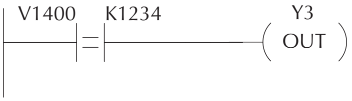
You can combine the various types of series and parallel branches to solve almost any application problem. The following example shows a simple combination network.



Comparative Boolean

The Comparative Boolean provides evaluation of two BCD values using boolean contacts. The valid evaluations are: equal to, not equal to, equal to or greater than, and less than.

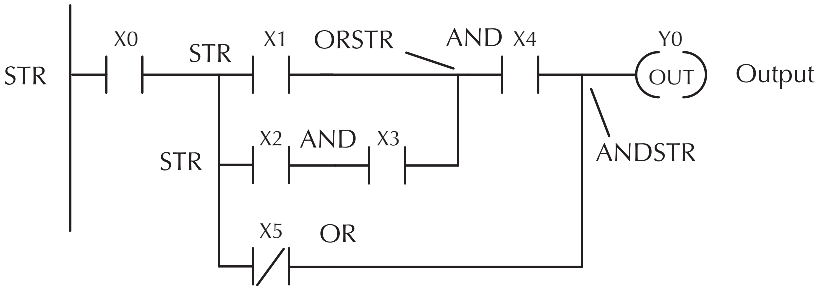
In this example, when the BCD value in V-memory location V1400 is equal to the constant value 1234, Y3 will energize.



Boolean Stack

There are limits to how many elements you can include in a rung. This is because the D4-454 PLCs 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 the program encounters a STR instruction, the instruction is placed on the top of the stack. Any other STR instructions already on the boolean stack are pushed down a level. The ANDSTR, and ORSTR instructions combine levels of the boolean stack when they are encountered. An error will occur during program compilation 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	X2 AND X3
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	

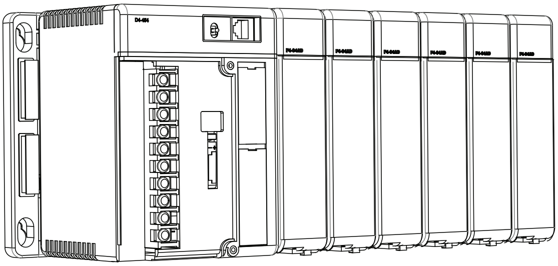
ANDSTR

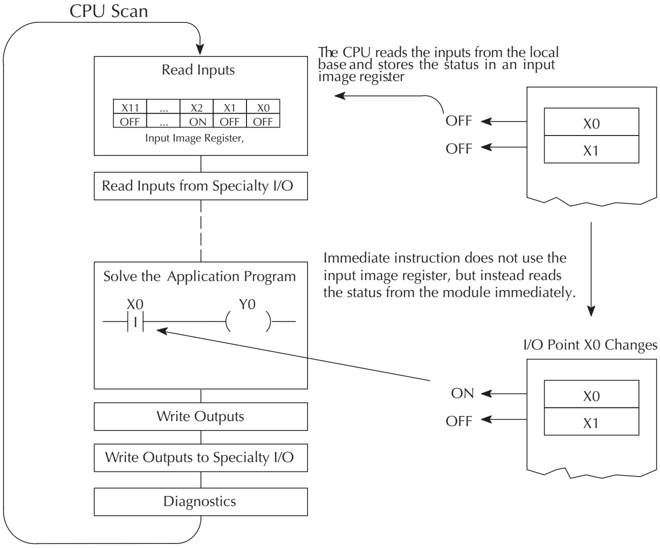
1	XO AND (NOT X5 or X4) AND {X1 or (X2 AND X3)}
2	
3	

Immediate Boolean

The D4-454 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 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.*

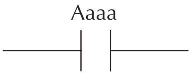



CPU Scan
Read Inputs
X11 | ... | X2 | X1 | X0
OFF | ... | ON | OFF | OFF
Input Image Register,
Read Inputs from Specialty I/O
Solve the Application Program
X0
I
Y0
Write Outputs
Write Outputs to Specialty I/O
Diagnostics
The CPU reads the inputs from the local base and stores the status in an input image register
OFF
OFF
X0
X1
Immediate instruction does not use the input image register, but instead reads the status from the module immediately.
I/O Point X0 Changes
ON
OFF
X0
X1

Boolean Instructions

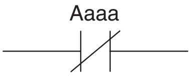
Store (STR)

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.



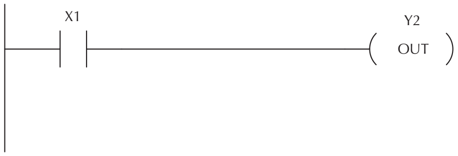
Store Not (STRN)

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

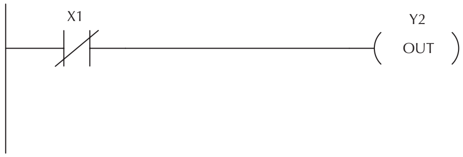


Operand Data Type		D4-454 Range
	A	aaa
Inputs	X	0 – 1777
Outputs	Y	0 – 1777
Control Relays	C	0 – 3777
Stage	S	0 – 1777
Timer	T	0 – 377
Counter	CT	0 – 377
Special Relay	SP	0 – 777

In the following Store example, when input X1 is on, output Y2 will energize.

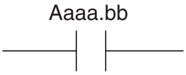


In the following Store Not example, when input X1 is off output Y2 will energize.



Store Bit-of-Word (STRB)

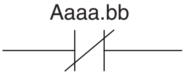
The Store Bit-of-Word instruction begins a new rung or an additional branch in a rung with a normally open contact. Status of the contact will be the same state as the bit referenced in the associated memory location.



Store Not Bit-of-Word (STRNB)

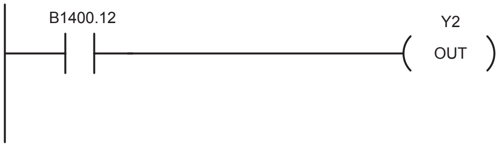
The Store Not instruction begins a new rung or an additional branch in a rung with a normally closed contact. Status of the contact will be opposite the state of the bit referenced in the associated memory location.

These instructions look like the STR and STRN instructions only the address is different. Take note how the address is set up in the following Store Bit-of-Word example.

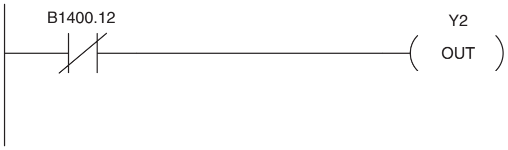


Operand Data Type		D4-454 Range	
	A	aaa	bb
V-memory	B	See memory map	0 to 15
Pointer	PB	See memory map	0 to 15

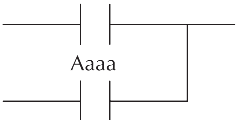
When bit 12 of V-memory location V1400 is on, output Y2 will energize.



In the following Store Not Bit-of-Word example, when bit 12 of V-memory location V1400 is off, output Y2 will energize.

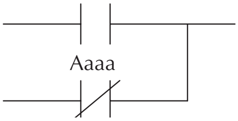


Or (OR)



The Or instruction will logically OR 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.

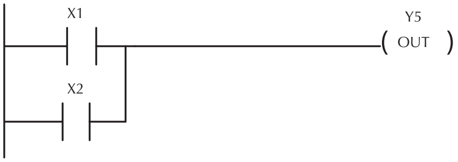
Or Not (ORN)



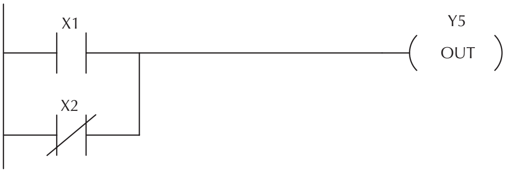
The Or Not instruction will logically OR a normally closed contact in parallel with another contact in a rung. The status of the contact will be opposite the state of the associated image register point or memory location.

Operand Data Type		D4-454 Range
	A	aaa
Inputs	X	0 – 1777
Outputs	Y	0 – 1777
Control Relays	C	0 – 3777
Stage	S	0 – 1777
Timer	T	0 – 377
Counter	CT	0 – 377
Special Relay	SP	0 – 777

In the following Or example, when input X1 or X2 is on, output Y5 will energize.

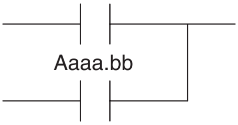


In the following Or Not example, when input X1 is on or X2 is off, output Y5 will energize.



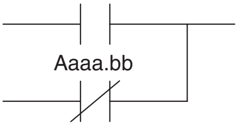
Or Bit-of-Word (OR)

The Or Bit-of-Word instruction will logically OR a normally open 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.



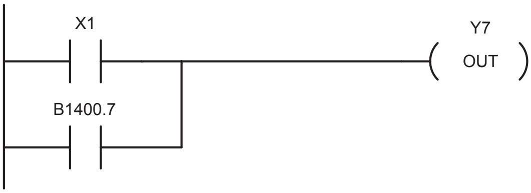
Or Not Bit-of-Word (ORN)

The Or Not Bit-of-Word instruction will logically OR a normally closed 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.

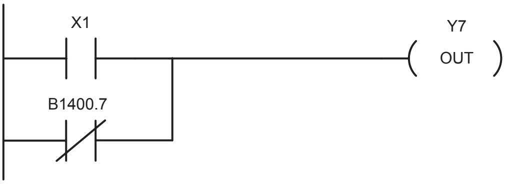


Operand Data Type		D4-454 Range	
	A	aaa	bb
V-memory	B	See memory map	0 to 15
Pointer	PB	See memory map	0 to 15

In the following Or Bit-of-Word example, when input X1 or bit 7 of V1400 is on, output Y5 will energize.

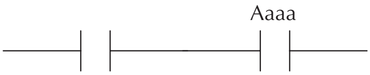


In the following Or Bit-of-Word example, when input X1 is on or bit 7 of V1400 is off, output Y7 will energize.



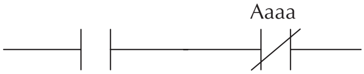
AND (AND)

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.



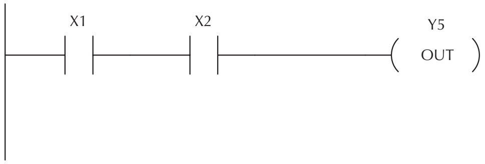
AND NOT (ANDN)

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.

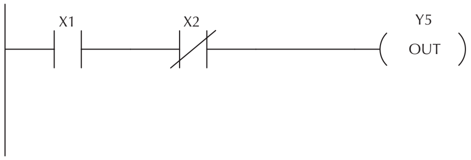


Operand Data Type		D4-454 Range
	A	aaa
Inputs	X	0 – 1777
Outputs	Y	0 – 1777
Control Relays	C	0 – 3777
Stage	S	0 – 1777
Timer	T	0 – 377
Counter	CT	0 – 377
Special Relay	SP	0 – 777

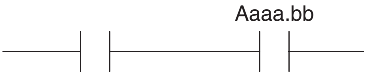
In the following And example, when input X1 and X2 are on output Y5 will energize.



In the following And Not example, when input X1 is on and X2 is off output Y5 will energize.

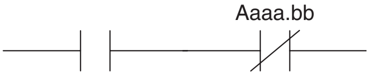


AND Bit-of-Word (AND)



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.

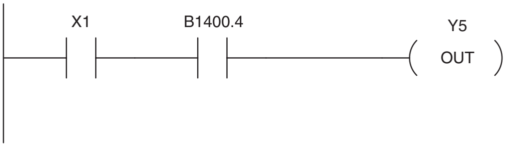
AND Not Bit-of-Word (ANDN)



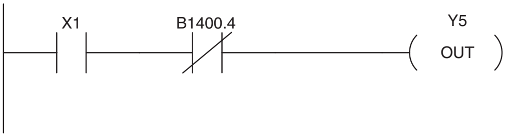
The And Not Bit-of-Word instruction logically ands a normally closed contact in series with another contact in a rung. The status of the contact will be opposite the state of the bit referenced in the associated memory location.

Operand Data Type		D4-454 Range	
	A	aaa	bb
V-memory	B	See memory map	0 to 15
Pointer	PB	See memory map	0 to 15

In the following And Bit-of-Word example, when input X1 and bit 4 of V1400 is on output Y5 will energize.

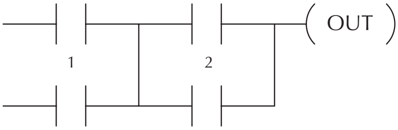


In the following And Not Bit-of-Word example, when input X1 is on and bit 4 of V1400 is off output Y5 will energize.



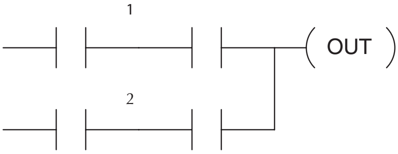
And Store (ANDSTR)

The And Store instruction logically ands two branches of a rung in series. Both branches must begin with the Store instruction.

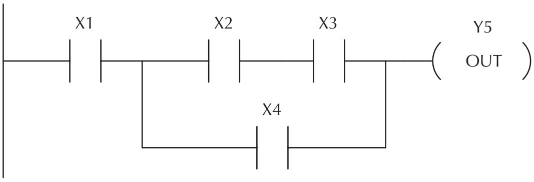


OR Store (ORSTR)

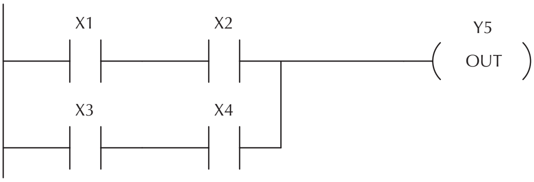
The OR Store instruction logically ORs two branches of a rung in parallel. Both branches must begin with the Store instruction.



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.

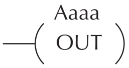


In the following OR Store example, the branch consisting of X1 and X2 have been OR'd with the branch consisting of X3 and X4.



Out (OUT)

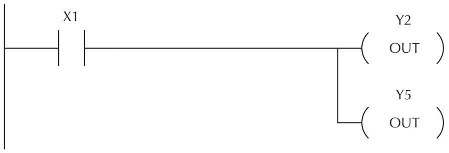
The Out instruction reflects the status of the rung (on/off) and outputs the discrete (on/off) state to the specified image register point or memory location.



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.

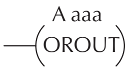
Operand Data Type		D4-454 Range
	A	aaa
Inputs	X	0 – 1777
Outputs	Y	0 – 1777
Control Relays	C	0 – 3777

In the following Out example, when input X1 is on, output Y2 and Y5 will energize.



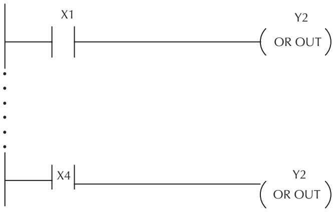
Or Out (OROUT)

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.



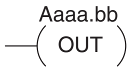
Operand Data Type		D4-454 Range
	A	aaa
Inputs	X	0– 1777
Outputs	Y	0 – 1777
Control Relays	C	0 – 3777

In the following example, when X1 or X4 is on, Y2 will energize.



Out Bit-of-Word (OUT)

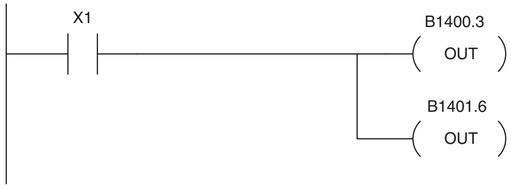
The OUT Bit-of-Word instruction reflects the status of the rung (on/off) and outputs the discrete (on/off) state to the specified bit in the referenced memory location. Multiple Out Bit-of-Word instructions referencing the same bit of the same word generally should not be used since only the last OUT instruction in the program will control the status of the bit.



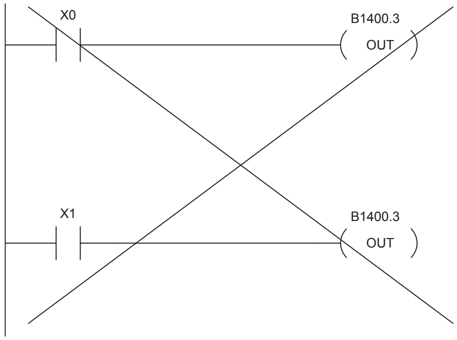
Operand Data Type		D4-454 Range	
	A	aaa	bb
V-memory	B	See memory map	0 to 15
Pointer	PB	See memory map	0 to 15

NOTE: *If the Bit-of-Word is entered as V1400.3 in DirectSOFT, it will be converted to B1400.3. Bit-of-Word can also be entered as B1400.3.*

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.

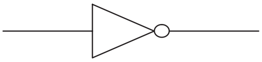


The following Out Bit-of-Word example contains two Out Bit-of-Word instructions using the same bit in the same memory word. The final state bit 3 of V1400 is ultimately controlled by the last rung of logic referencing it. X1 will override the logic state controlled by X0. To avoid this situation, multiple outputs using the same location must not be used in programming.

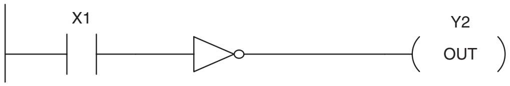


Not (NOT)

The NOT instruction inverts the status of the rung at the point of the instruction.

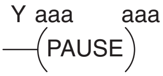


In the following example, when X1 is off, Y2 will energize. This is because the NOT instruction inverts the status of the rung at the NOT instruction.



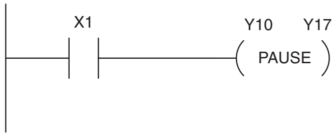
Pause (PAUSE)

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 module.



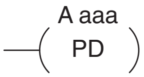
In the following example, when X1 is ON, Y1-Y17 will be turned OFF at the output module. The execution of the ladder program will not be affected.

Operand Data Type		D4-454 Range
	A	aaa
Outputs	Y	0–1777



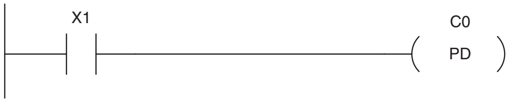
Positive Differential (PD)

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.



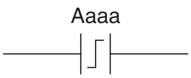
In the following example, every time X1 makes an Off-to-On transition, C0 will energize for one scan.

Operand Data Type		D4-454 Range
	A	aaa
Inputs	X	0–1777
Outputs	Y	0–1777
Control Relays	C	0–3777



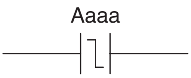
Store Positive Differential (STRPD)

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.



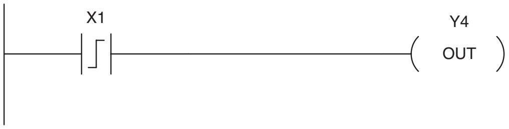
Store Negative Differential (STRND)

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).

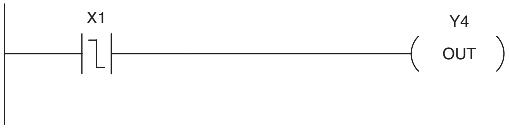


Operand Data Type		D4-454 Range
	A	aaa
Inputs	X	0 – 1777
Outputs	Y	0 – 1777
Control Relays	C	0 – 3777
Stage	S	0 – 1777
Timer	T	0 – 377
Counter	CT	0 – 377

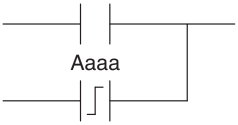
In the following example, each time X1 makes an Off-to-On transition, Y4 will energize for one scan.



In the following example, each time X1 makes an On-to-Off transition, Y4 will energize for one scan.

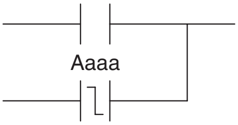


Or Positive Differential (ORPD)



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.

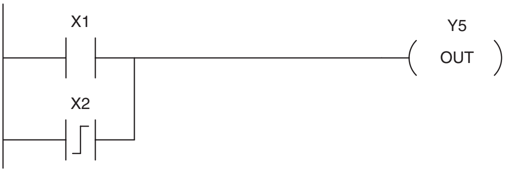
Or Negative Differential (ORND)



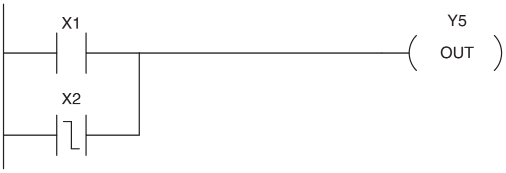
The Or Negative Differential instruction logically ORs a contact in parallel with another contact in a rung. The status of the contact will be open until the associated image register point makes an On-to-Off transition, closing it for one CPU scan. Thereafter, it remains open until another On-to-Off transition.

Operand Data Type		D4-454 Range
	A	aaa
Inputs	X	0 – 1777
Outputs	Y	0 – 1777
Control Relays	C	0 – 3777
Stage	S	0 – 1777
Timer	T	0 – 377
Counter	CT	0 – 177

In the following example, Y5 will energize whenever X1 is on, or for one CPU scan when X2 transitions from Off to On.

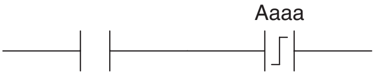


In the following example, Y5 will energize whenever X1 is on, or for one CPU scan when X2 transitions from On to Off.



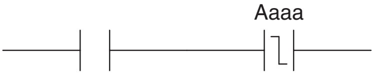
And Positive Differential (ANDPD)

The And Positive Differential instruction logically ANDs a normally open 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.



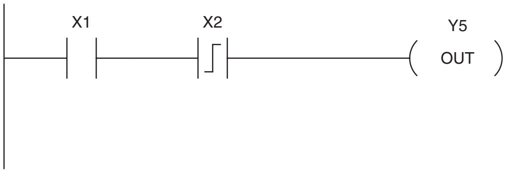
And Negative Differential (ANDND)

The And Negative Differential instruction logically ANDs a normally open contact in series with another contact 5-22 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.

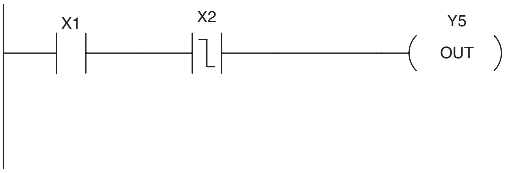


Operand Data Type		D4-454 Range
	A	aaa
Inputs	X	0–1777
Outputs	Y	0–1777
Control Relays	C	0–3777
Stage	S	0–1777
Timer	T	0–377
Counter	CT	0–377

In the following example, Y5 will energize for one CPU scan whenever X1 is on and X2 transitions from Off to On.

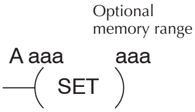


In the following example, Y5 will energize for one CPU scan whenever X1 is on and X2 transitions from On to Off.



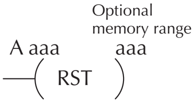
Set (SET)

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.



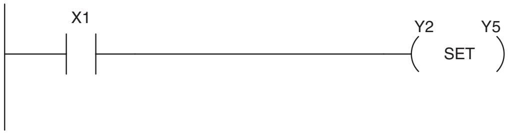
Reset (RST)

The Reset instruction reset,s 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.

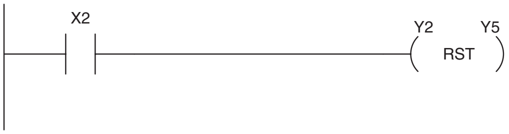


Operand Data Type		D4-454 Range
	A	aaa
Inputs	X	0 – 1777
Outputs	Y	0 – 1777
Control Relays	C	0 – 3777
Stage	S	0 – 1777
Timer	T	0 – 377
Counter	CT	0 – 377

In the following example when X1 is on, Y2 through Y5 will energize.

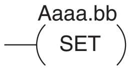


In the following example when X2 is on, Y2 through Y5 will be reset or de-energized.



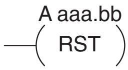
Set Bit-of-Word (SET)

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.



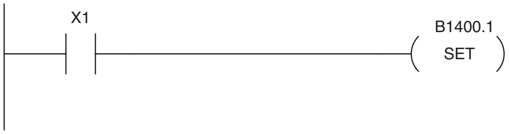
Reset Bit-of-Word (RST)

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.

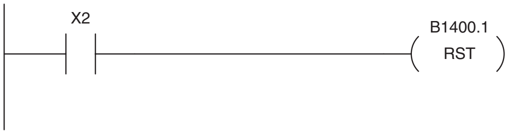


Operand Data Type		D4-454 Range	
	A	aaa	bb
V-memory	B	See memory map	0 to 15
Pointer	PB	See memory map	0 to 15

In the following example, when X1 turns on, bit 1 in V1400 is set to the on state.

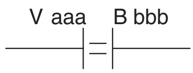


In the following example, when X2 turns on, bit 1 in V1400 is reset to the off state.



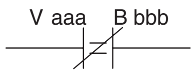
Comparative Boolean

Store If Equal (STRE)



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 Vaaa is equal to Bbbb.

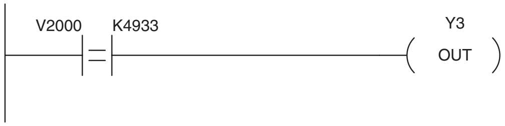
Store If Not Equal (STRNE)



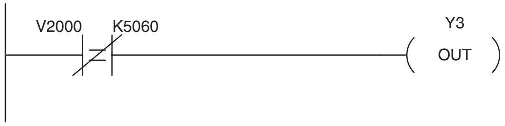
The Store If Not Equal instruction begins a new rung or additional branch in a rung with a normally closed comparative contact. The contact will be on when Vaaa does not equal Bbbb

Operand Data Type		D4-454 Range	
	B	aaa	bbb
V-memory	V	See memory map	See memory map
Pointer	P	See memory map	See memory map
Constant	K	– –	0 – 9999

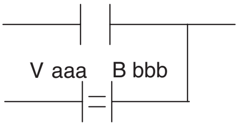
In the following example, when the BCD value in V-memory location V2000 is equal to 4933, Y3 will energize.



In the following example, when the BCD value in V-memory location V2000 does not equal 5060, Y3 will energize.

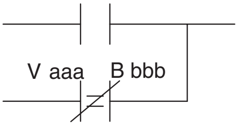


Or If Equal (ORE)



The Or If Equal instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Vaaa is equal to Bbbb.

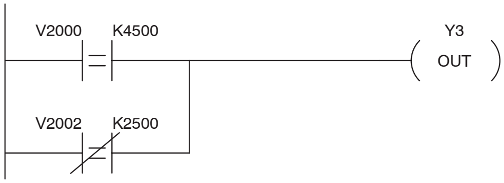
Or If Not Equal (ORNE)



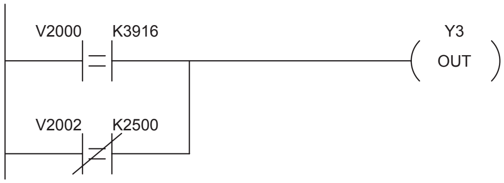
The Or If Not Equal instruction connects a normally closed comparative contact in parallel with another contact. The contact will be on when Vaaa does not equal Bbbb.

Operand Data Type		D4-454 Range	
	B	aaa	bbb
V-memory	V	See memory map	See memory map
Pointer	P	See memory map	See memory map
Constant	K	– –	0 – 9999

In the following example, when the BCD value in V-memory location V2000 is equal to 4500 or V2002 does not equal 2500, Y3 will energize.

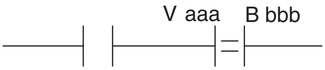


In the following example, when the BCD value in V-memory location V2000 is equal to 3916   or V2002 does not equal 2500, Y3 will energize



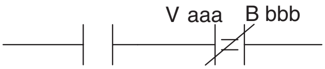
And If Equal (ANDE)

The And If Equal instruction connects a normally open comparative contact in series with another contact. The contact will be on when Vaaa is equal to Bbbb.



And If Not Equal (ANDNE)

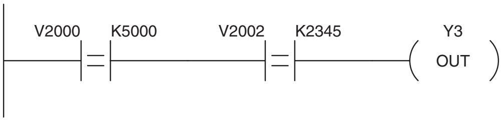
The And If Not Equal instruction connects a normally closed comparative contact in series with another contact. The contact will be on when Vaaa does not equal Bbb.

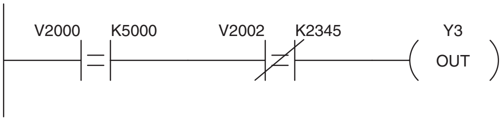


Operand Data Type		D4-454 Range	
	B	aaa	bbb
V-memory	V	See memory map	See memory map
Pointer	P	See memory map	See memory map
Constant	K	– –	0 – 9999

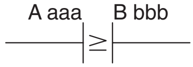
In the following example, when the BCD value in V-memory location V2000 is equal to 5000 and V2002 is equal to 2345, Y3 will energize.

In the following example, when the BCD value in V-memory location V2000 is equal to 5000 and V2002 does not equal 2345, Y3 will energize.



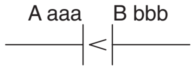


Store (STR)



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.

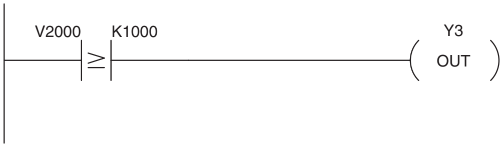
Store Not (STRN)



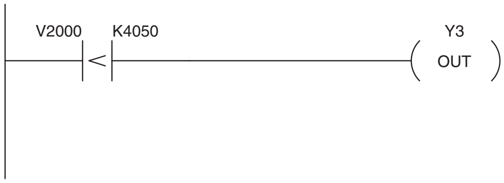
The Comparative Store Not instruction begins a new rung or additional branch in a rung with a normally closed comparative contact. The contact will be on when Aaaa is less than Bbbb.

Operand Data Type		D4-454 Range	
	A/B	aaa	bbb
V-memory	V	See memory map	See memory map
Pointer	p	See memory map	See memory map
Constant	K	– –	0 – 9999
Timer	TA	0 – 377	
Counter	CTA	0 – 377	

In the following example, when the BCD value in V-memory location V2000 is equal to or greater than 1000, Y3 will energize.

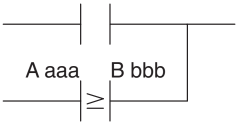


In the following example, when the value in V-memory location V2000 is less than 4050, Y3 will energize.



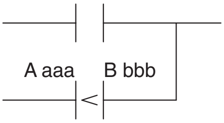
Or (OR)

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.



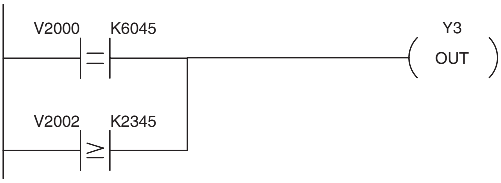
Or Not (ORN)

The Comparative Or Not instruction connects a normally closed comparative contact in parallel with another contact. The contact will be on when Aaaa is less than Bbbb.

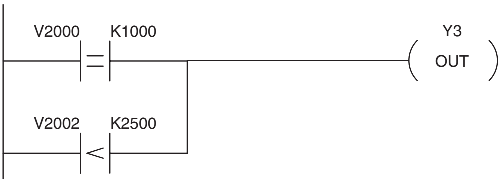


Operand Data Type		D4-454 Range	
	A/B	aaa	bbb
V-memory	V	See memory map	See memory map
Pointer	p	See memory map	See memory map
Constant	K	– –	0 – 9999
Timer	TA	0 – 377	
Counter	CTA	0 – 377	

In the following example, when the BCD value in V-memory location V2000 is equal to 6045 or V2002 is equal to or greater than 2345, Y3 will energize.

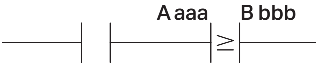


In the following example when the BCD value in V-memory location V2000 is equal to 1000 or V2002 is less than 2500, Y3 will energize.



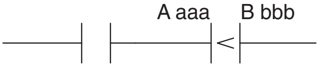
And (AND)

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.



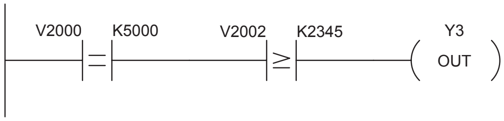
And Not (ANDN)

The Comparative And Not instruction connects a normally closed comparative contact in series with another contact. The contact will be on when Aaaa is less than Bbbb.

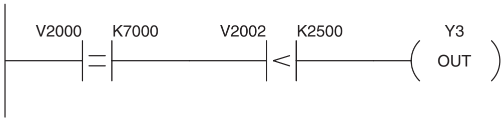


Operand Data Type		D4-454 Range	
	A/B	aaa	bbb
V-memory	V	See memory map	See memory map
Pointer	p	See memory map	See memory map
Constant	K	– –	0 – 9999
Timer	TA	0 – 377	
Counter	CTA	0 – 377	

In the following example, when the value in BCD V-memory location V2000 is equal to 5000, and V2002 is equal to or greater than 2345, Y3 will energize.

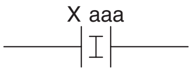


In the following example, when the value in V-memory location V2000 is equal to 7000 and V2002 is less than 2500, Y3 will energize.



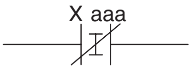
Immediate Instructions

Store Immediate (STRI)



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.

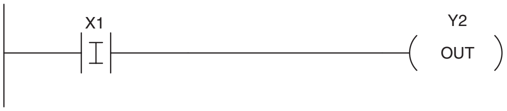
Store Not Immediate (STRNI)



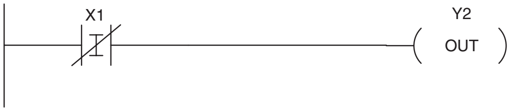
The Store Not Immediate instruction begins a new rung or additional branch in a rung. The status of the contact will be opposite the status of the associated input point at the time the instruction is executed. The image register is not update

Operand Data Type		D4-454 Range
		aaa
Inputs	X	0 – 1777

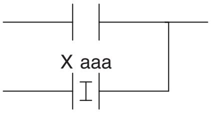
In the following example, when X1 is on, Y2 will energize.



In the following example, when X1 is off, Y2 will energize.

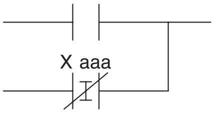


Or Immediate (ORI)



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.

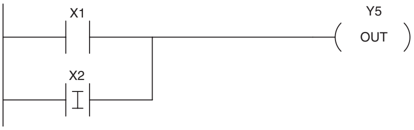
Or Not Immediate (ORNI)



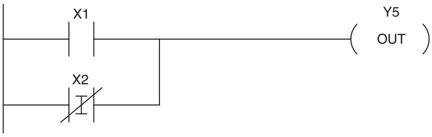
The Or Not Immediate connects two contacts in parallel. The status of the contact will be opposite the status of the associated input point at the time the instruction is executed. The image register is not updated.

Operand Data Type		D4-454 Range
		aaa
Inputs	X	0–1777

In the following example, when X1 or X2 is on, Y5 will energize.

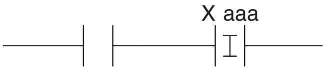


In the following example, when X1 is on or X2 is off, Y5 will energize.



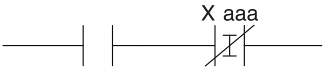
And Immediate (ANDI)

The And Immediate instruction 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.



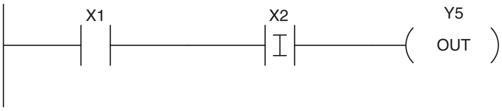
And Not Immediate (ANDNI)

The And Not Immediate instruction 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.

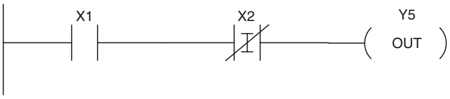


Operand Data Type		D4-454 Range
		aaa
Inputs	X	0–1777

In the following example, when X1 and X2 are on, Y5 will energize.

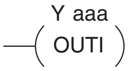


In the following example, when X1 is on and X2 is off, Y5 will energize.



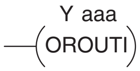
Out Immediate (OUTI)

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.



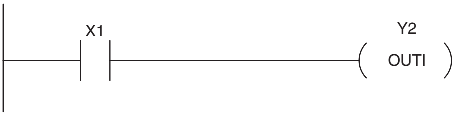
Or Out Immediate (OROUTI)

The Or Out Immediate instruction has been designed to use more than 1 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 OR'd together. If the status of any rung is on at the time the instruction is executed, the output will also be on.

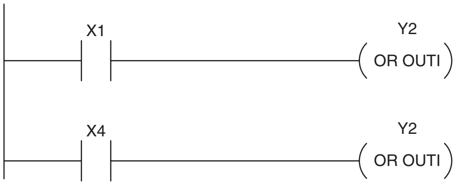


Operand Data Type		D4-454 Range
		aaa
Outputs	Y	0–1777

In the following example, when X1 is on, output point Y2 on the output module will turn on. .

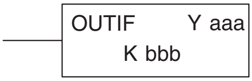


In the following example, when X1 or X4 is on, Y2 will energize.



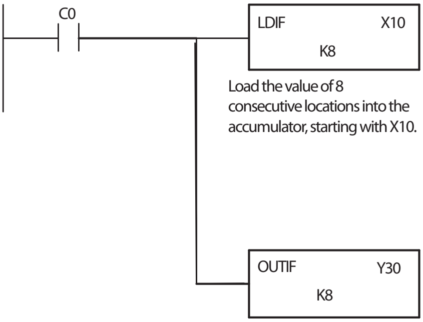
Out Immediate Formatted (OUTIF)

The Out Immediate Formatted instruction outputs a 1–32 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.



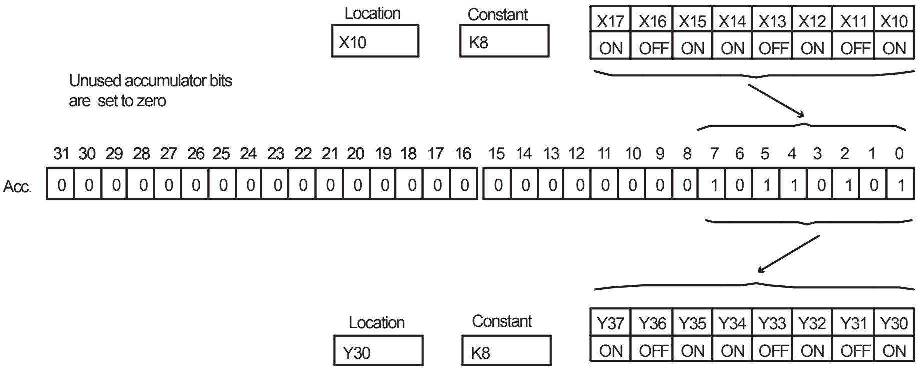
Operand Data Type		D4-454 Range
		aaa
Outputs	Y	0 – 1777
Constant	K	1 – 32

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)



Load the value of 8 consecutive locations into the accumulator, starting with X10.

Copy the value in the lower 8 bits of the accumulator to Y30-Y37



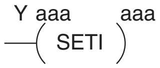
X17	X16	X15	X14	X13	X12	X11	X10
ON	OFF	ON	ON	OFF	ON	OFF	ON

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	1

Y37	Y36	Y35	Y34	Y33	Y32	Y31	Y30
ON	OFF	ON	ON	OFF	ON	OFF	ON

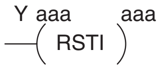
Set Immediate (SETI)

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. The image register is not updated. 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.



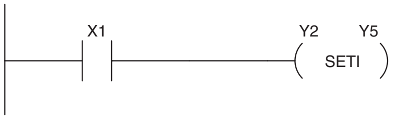
Reset Immediate (RSTI)

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. The image register is not updated. Once the outputs are reset, it is not necessary for the input to remain on.

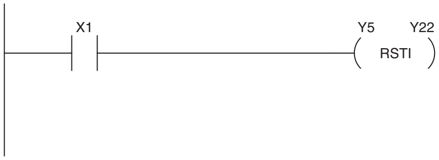


Operand Data Type		DL454 Range
		aaa
Outputs	Y	0–1777

In the following example, when X1 is on, Y2 through Y5 will be set (on) for the corresponding output points.

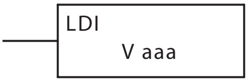


In the following example, when X1 is on, Y5 through Y22 will be reset (off) for the corresponding output module(s).



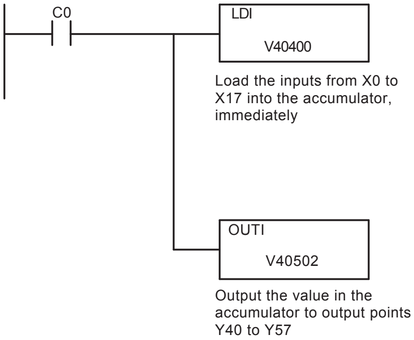
Load Immediate (LDI)

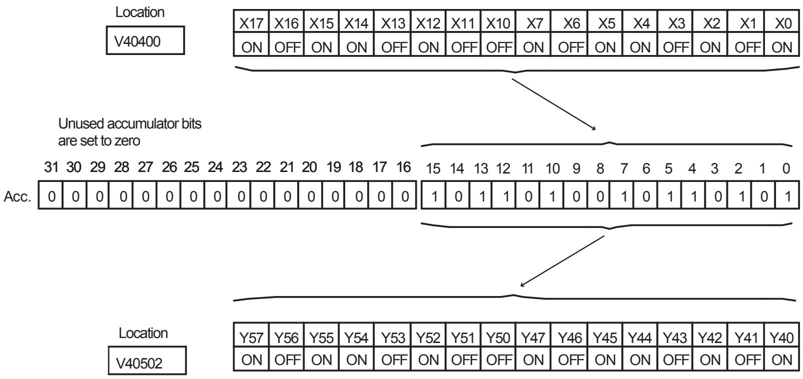
The Load Immediate instruction loads a 16-bit V-memory value into the accumulator. The valid address range includes all input point addresses on the local base. The value reflects the current status of the input points at the time the instruction is executed. This instruction may be used instead of the LDIF instruction, which requires you to specify the number of input points.



Operand Data Type		D4-454 Range
		aaa
Inputs	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).


C0
LDI
V40400
Load the inputs from X0 to X17 into the accumulator, immediately
OUTI
V40502
Output the value in the accumulator to output points Y40 to Y57



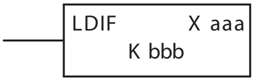
X17	X16	X15	X14	X13	X12	X11	X10	X7	X6	X5	X4	X3	X2	X1	X0
ON	OFF	ON	ON	OFF	ON	OFF	OFF	ON	OFF	ON	ON	OFF	ON	OFF	ON

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0	1

Y57	Y56	Y55	Y54	Y53	Y52	Y51	Y50	Y47	Y46	Y45	Y44	Y43	Y42	Y41	Y40
ON	OFF	ON	ON	OFF	ON	OFF	OFF	ON	OFF	ON	ON	OFF	ON	OFF	ON

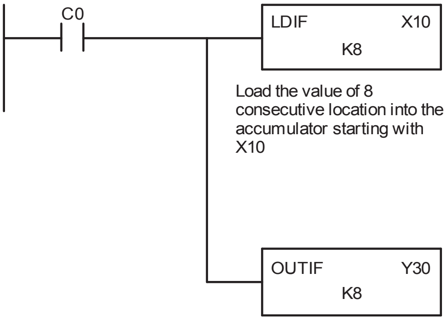
Load Immediate Formatted (LDIF)

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.



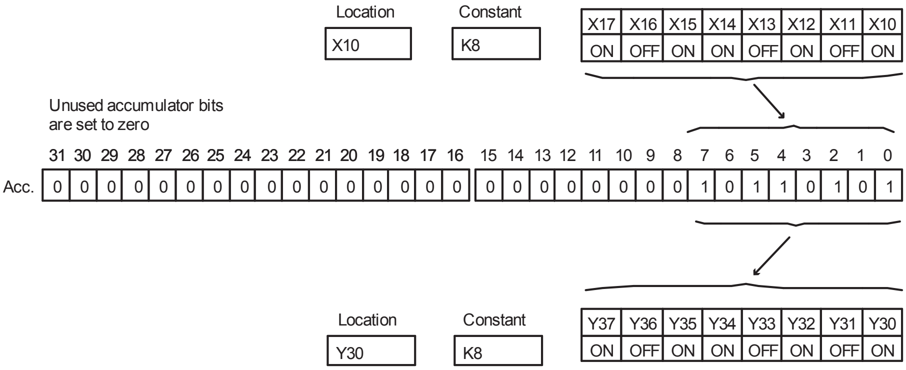
Operand Data Type		D4-454 Range	
		aaa	bbb
Inputs	X	0 – 1777	- -
Constant	K	- -	1-32

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).



Load the value of 8 consecutive location into the accumulator starting with X10

Copy the value of the lower 8 bits of the accumulator to Y30 -Y37



X17	X16	X15	X14	X13	X12	X11	X10
ON	OFF	ON	ON	OFF	ON	OFF	ON

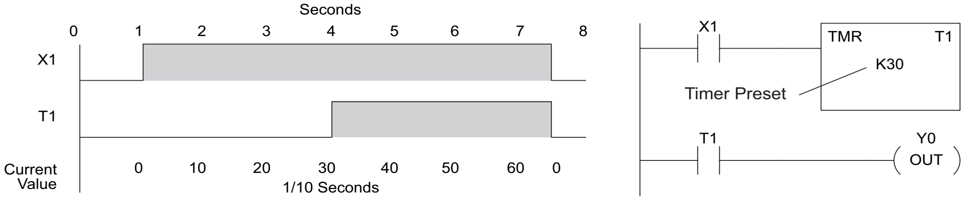
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	1

Y37	Y36	Y35	Y34	Y33	Y32	Y31	Y30
ON	OFF	ON	ON	OFF	ON	OFF	ON

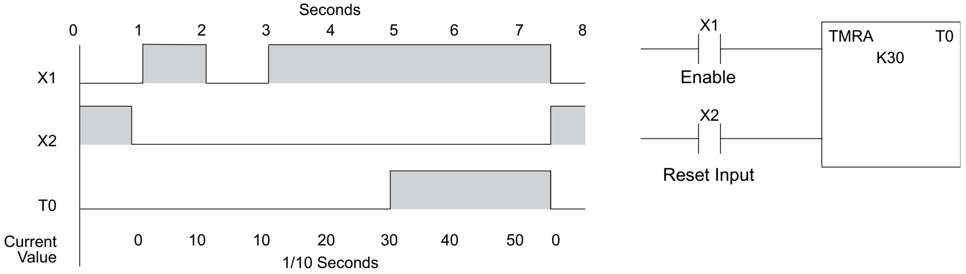
Timer, Counter and Shift Register Instructions

Using Timers

Timers are used to time an event for a desired period. The single input timer 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. There is a discrete bit 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.

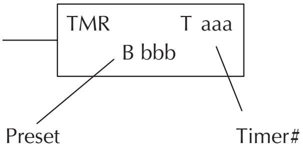


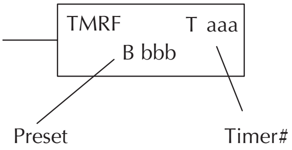
There are those applications that need an accumulating timer, meaning it has the ability to time, stop, and then resume from where it previously stopped. The accumulating timer works similarly to the regular timer, but two inputs are required. The enable input starts and stops the timer. When the timer 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. There is a tenth of a second and a hundredth of a second timer 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.



NOTE: *Decimal points are not used in these timers, but the decimal point is implied. The preset and current value for all four timers is in BCD format.*

Timer (TMR) and Timer Fast (TMRF)





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). Both timers use single word BCD values for the preset and current value. The decimal place is implied.

Instruction Specifications

Timer Reference (Taaa): Specifies the timer number.

Preset Value (Bbbb): Constant value (K) or a V-memory location specified in BCD.

Current Value: Timer current values, in BCD format, 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. Operating as a "timer done bit", 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 is T2.

NOTE: *A V-memory preset is required when the ladder program or an Operator Interface unit is used to change the preset.*

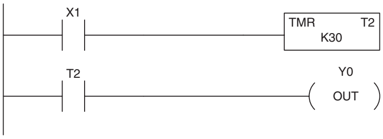
Operand Data Type		D4-454 Range	
	A/B	aaa	bbb
Timers	T	0–377	– –
V-memory for preset values	V	– –	1400–7377 10000-17777
Pointers (preset only)	P	– –	1400–7377 10000-17777
Constants (preset only)	K	– –	0–9999
Timer discrete status bits	T/V	0–377 or V41100–41117	
Timer current values	V /T**	0–377	

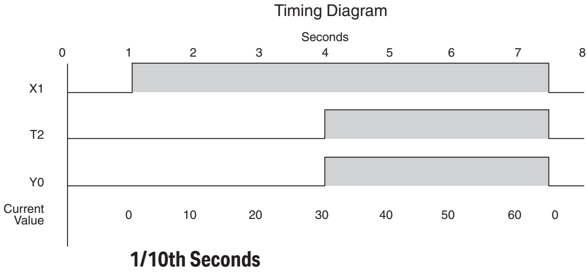
NOTE: *The DirectSOFT programming uses separate references, such as "T2" for discrete status bit for Timer T2, and "TA2" for the current value of Timer T2.*

You can perform functions when the timer reaches the specified preset using the discrete status bit. Or, use comparative contacts to perform functions at different time intervals, based on one timer. The examples on the following page show these two methods of programming timers.

Timer Example Using Discrete Status Bits

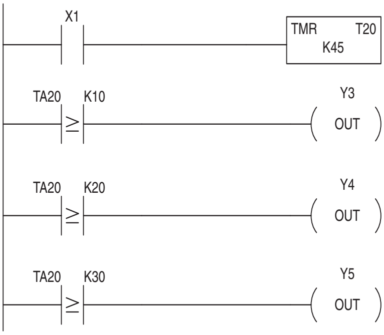
In the following example, a single input timer is used with a preset of 3 seconds. The timer discrete status bit (T2) will turn on when the timer has timed for 3 seconds. The timer is reset when X1 turns off, turning the discrete status bit off and resetting the timer current value to 0.

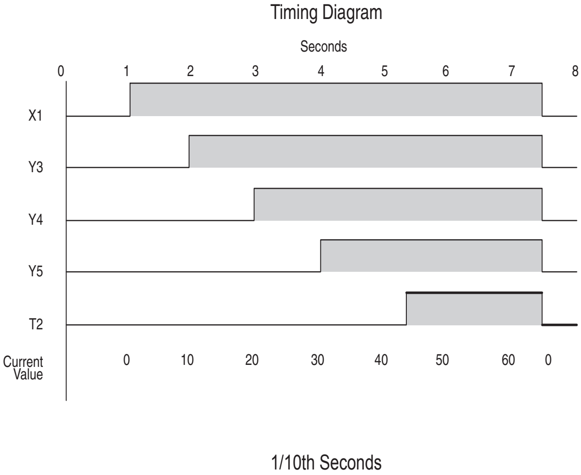




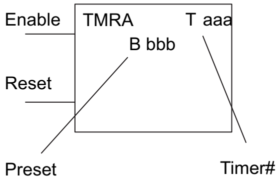
Timer Example Using Comparative Contacts

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.



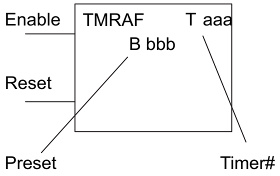


Accumulating Timer (TMRA)



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)



The Accumulating Fast Timer is a 0.01 second two-input timer that will time to a maximum of 999999.99. The TMRA uses two timer registers in V-memory.

Each timer uses two timer registers in V-memory. The preset and current values are in double word BCD format, and the decimal point is implied. These timers have two inputs, an enable and a reset. The timer starts timing when the enable is on and stops when the enable is off (without resetting the count). The reset will reset the timer when on and allow the timer to time when off.

Timer Reference (Taaa): Specifies the timer number.

Preset Value (Bbbb): Constant value (K) or V-memory.

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, V3.

Discrete Status Bit: The discrete status bit is accessed by referencing the associated T memory location. Operating as a "timer done bit," 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.

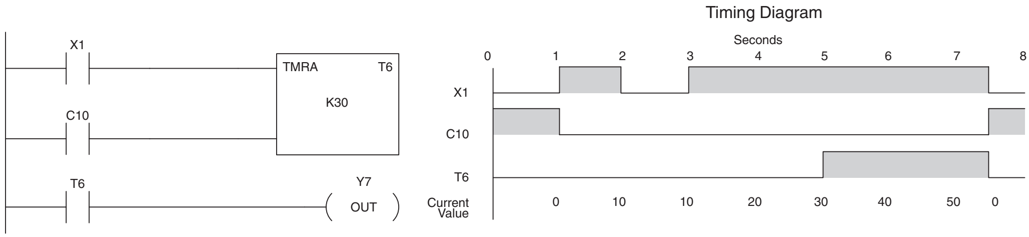
NOTE 1: *The accumulating timer uses two consecutive V-memory locations for the 8-digit value, therefore two consecutive timer locations. For example, if TMRA T1 is used, the next available timer number is T3*
NOTE 2: A V-Memory preset is required if the ladder program or an Operator Interface panel is used to change the preset.

Operand Data Type		D4-454 Range	
	A/B	aaa	bbb
Timers	T	0 – 376	– –
V-memory for preset values	V	– –	1400 – 7377 10000-37777
Pointers (preset only)	P	– –	1400 – 7377 10000-37777
Constants (preset only)	K	– –	0 – 99999999
Timer discrete status bits	T/V	0 – 377 or V41100 – 41117	
Timer current values	V /T**	0 – 377	

NOTE: *The DirectSOFT programming software uses separate references, such as "T2" for discrete status bit for Timer T2, and "TA2" for the current value of Timer T2.*

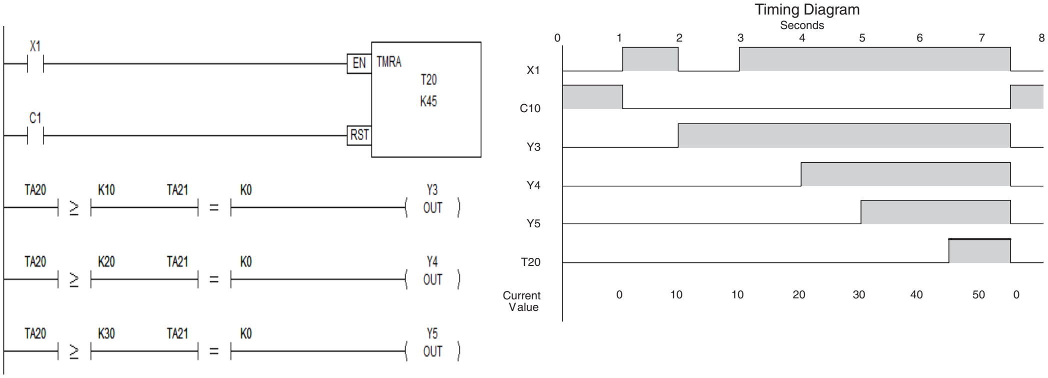
Accumulating Timer Example using Discrete Status Bits

In the following example, a two input timer (accumulating timer) is used with a preset of 3 seconds. The timer discrete status bit (T6) will turn on when the timer has timed for 3 seconds. Notice, in this example, that the timer times for 1 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 0.



Accumulator Timer Example Using Comparative Contacts

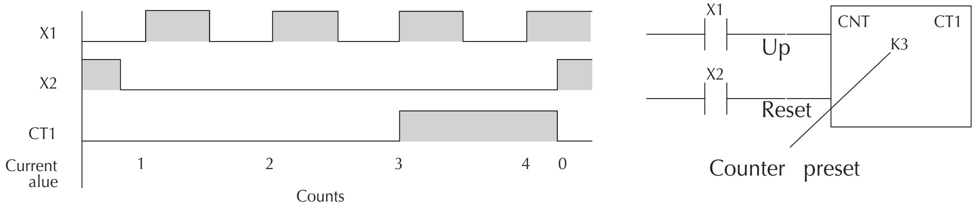
In the following example, a single input timer is used with a preset of 4.5 seconds. Comparative contacts are used to energized Y3, Y4, and Y5 at one second intervals respectively. The comparative contacts will turn off when the timer is reset.



Using Counters

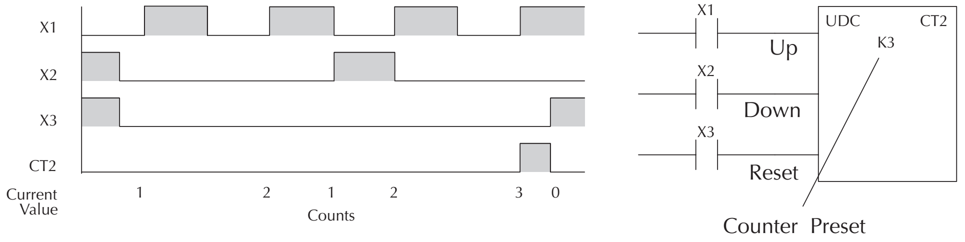
Counters are used to count events . The counters available are up counters, up/down counters, and stage counters (used with RLLPLUS programming).

The up counter (CNT) has two inputs, a count input and a reset input. The maximum count value is 9999. The timing diagram below shows the relationship between the counter input, counter reset, associated discrete bit, current value, and counter preset. The CNT counter preset and current value are both single word BCD values.

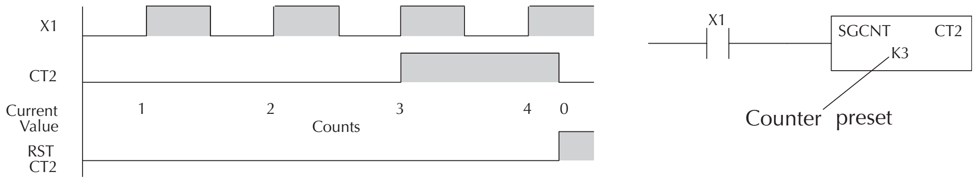


The up down counter (UDC) has three inputs, a count up input, count down input and reset input. The maximum count value is 99999999. The timing diagram below shows the relationship between the counter up and down inputs, counter reset, associated discrete bit, current value, and counter preset. The UDC counter preset and current value are both double word BCD values.

NOTE: *The UDC uses two consecutive V-memory locations for the 8-digit value, therefore, two consecutive timer locations. For example, if UDC CT1 is used, the next available counter number is CT3.*

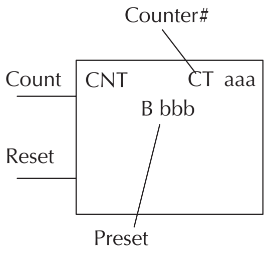


The stage counter (SGCNT) has a count input and is reset by the RST instruction. This instruction is useful when programming using the RLLPLUS structured programming. The maximum count value is 9999. The timing diagram below shows the relationship between the counter input, associated discrete bit, current value, counter preset and reset instruction.



Counter (CNT)

The Counter is a two-input counter that increments when the count input logic transitions from Off to On. When the counter reset input is On, the counter resets to 0. When the current value equals the preset value, the counter status bit comes 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.



Instruction Specifications

Counter Reference (CTaaa): Specifies the counter number.

Preset Value (Bbbb): Constant value (K) or a V-memory location.

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.

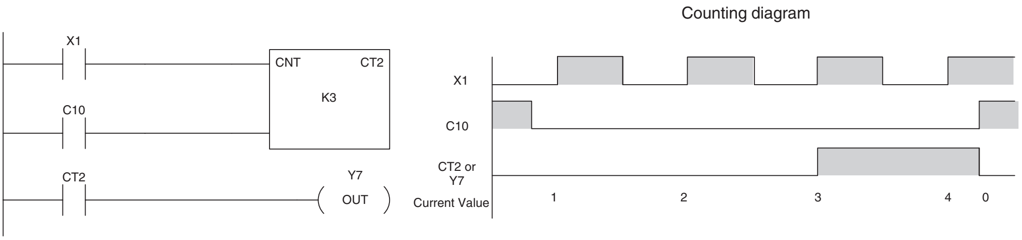
NOTE: *A V-memory preset is required if the ladder program or Operator Interface Panel is used to change the preset.*

Operand Data Type		D4-454 Range	
	A/B	aaa	bbb
Counters	CT	0 – 377	– –
V-memory (preset only)	V	– –	1400 – 7377 10000 – 37777
Pointers (preset only)	P	– –	1400 – 7377 10000 – 37777
Constants (preset only)	K	– –	0 – 9999
Counter discrete status bits	CT/V	0 – 377 or V41140 – V41157	
Counter current values	V /CT**	1000-1377	

NOTE: **May be non-volatile if MOV instruction is used. * DirectSOFT programming software uses separate references, such as “CT2” for discrete status bit for Counter CT2, and “CTA2” for the current value of Counter CT2.*

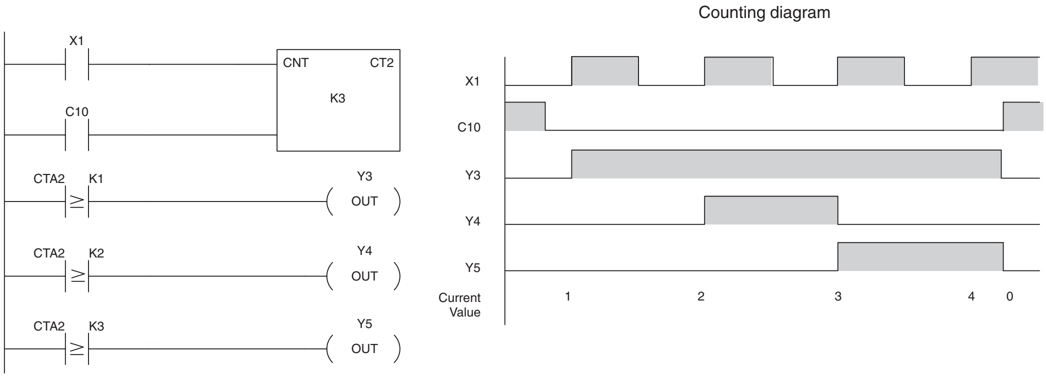
Counter Example Using Discrete Status Bits

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.

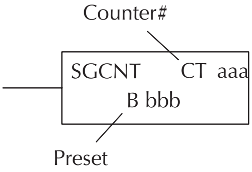


Counter Example Using Comparative Contacts

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.



Stage Counter (SGCNT)



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 RLLPLUS 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.

Instruction Specifications

Counter Reference (CTaaa): Specifies the counter number.

Preset Value (Bbbb): Constant value (K) or a V-memory location.

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.

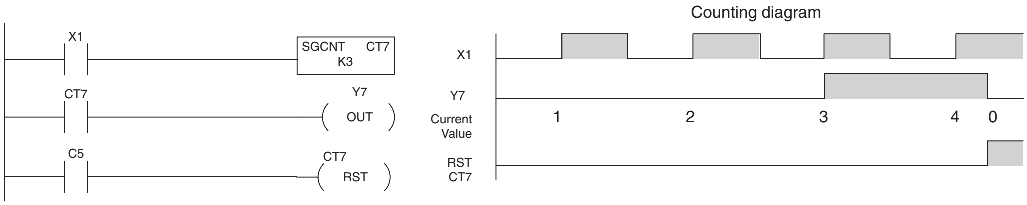
NOTE 1: *In using a counter inside a stage, the stage must be active for one scan before the input to the counter makes a 0-1 transition. Otherwise, there is no real transition and the counter will not count.*
NOTE 2: A V-memory preset is required only if the ladder program or an Operator Interface unit is used to change the preset.

Operand Data Type		D4-454 Range	
	A/B	aaa	bbb
Counters	CT	0 – 377	– –
V-memory (preset only)	V	– –	1400 – 7377 10000 – 37777
Pointers (preset only)	P	– –	1400 – 7377 10000 – 37777
Constants (preset only)	K	– –	0 – 9999
Counter discrete status bits	CT/V	0 – 377 or V41140 – V41157	
Counter current values	V /CT**	1000-1377	

NOTE: *The DirectSOFT programming software uses separate references, such as “CT2” for discrete status bit for Counter CT2, and “CTA2” for the current value of Counter CT2.*

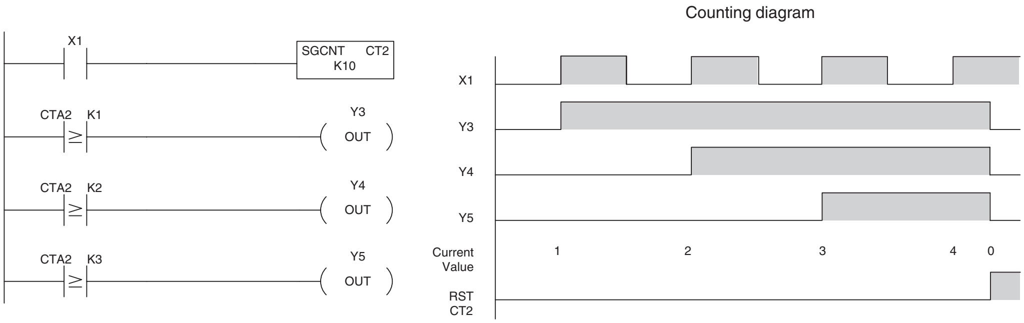
Stage Counter Example Using Discrete Status Bits

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.



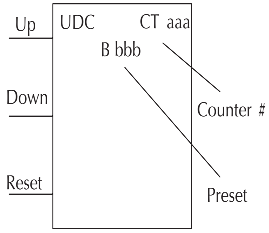
Stage Counter Example Using Comparative Contacts

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 (CTA2).



Up/Down Counter (UDC)

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 – 99999999. The count input not being used must be off in order for the active count input to function.



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.

Instruction Specification

Counter Reference (CTaaa): Specifies the counter number.

Preset Value (Bbbb): Constant value (K) or two consecutive V-memory locations, in BCD.

Current Values: Current count is a double word value accessed by referencing the associated V or CT memory locations in BCD. 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. Operating as a "counter done bit" 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.

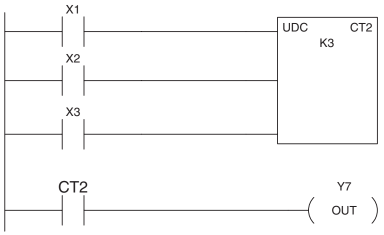
NOTE 1: *The UDC uses two consecutive V-memory locations for the 8-digit value, therefore two consecutive counter locations. For example, if UDC CT1 is used, the next available counter number is CT3.*
NOTE 2: *A V-memory preset is required only if the ladder program or an Operator Interface is used to change the preset.*

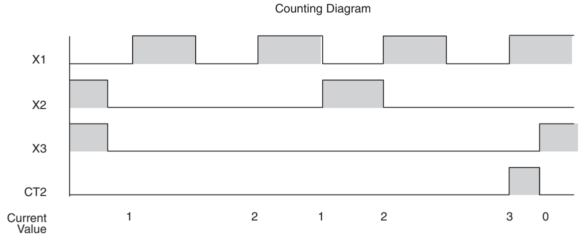
Operand Data Type		D4-454 Range	
	A/B	aaa	bbb
Counters	CT	0 – 376	– –
V-memory (preset only)	V	– –	1400 – 7377 10000 – 37777
Pointers (preset only)	P	– –	1400 – 7377 10000 – 37777
Constants (preset only)	K	– –	0 – 99999999
Counter discrete status bits	CT/V	0 – 377 or V41140 – V41157	
Counter current values	V /CT**	1000-1377	

NOTE: *The DirectSOFT programming software 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

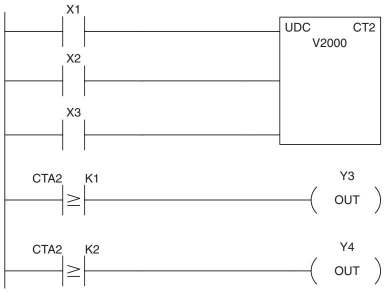
In the following example, if X2 and X3 are off, the counter will increment by one when X1 toggles from Off to On. If X1 and X3 are off, the counter will decrement by one when X2 toggles from Off to On. When the count value reaches the preset value of 3, the counter status bit will turn on. When the reset X3 turns on, the counter status bit will turn off and the current value will be 0.

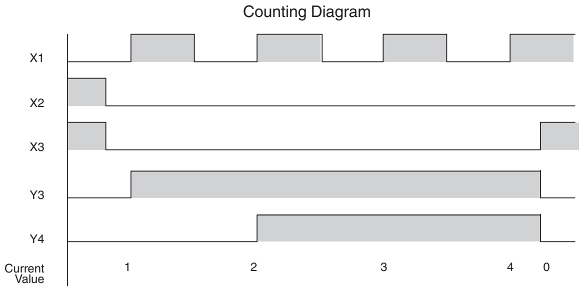



Counting Diagram

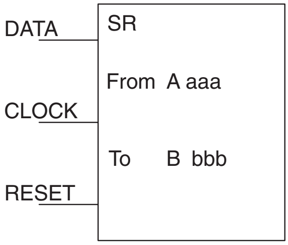
Up / Down Counter Example Using Comparative Contacts

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.




Counting Diagram

Shift Register (SR)



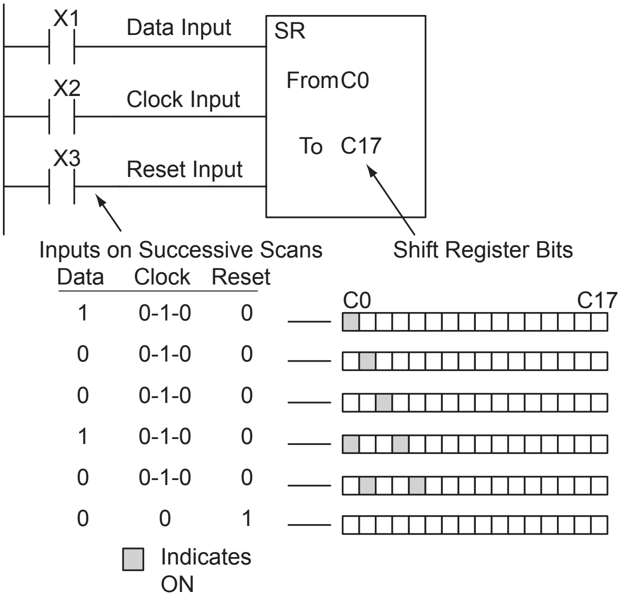
The Shift Register instruction shifts data through a predefined number of control relays. The control ranges in the shift register block must start at the beginning of an 8 bit boundary and must use 8-bit blocks.

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 sixteen bits to be shifted from left to right. From C17 to C0 would define a block of sixteen 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		D4-454 Range	
	A/B	aaa	bbb
Control Relay	C	0 – 3777	0 – 3777



Data	Clock	Reset
1	0-1-0	0
0	0-1-0	0
0	0-1-0	0
1	0-1-0	0
0	0-1-0	0
0	0	1

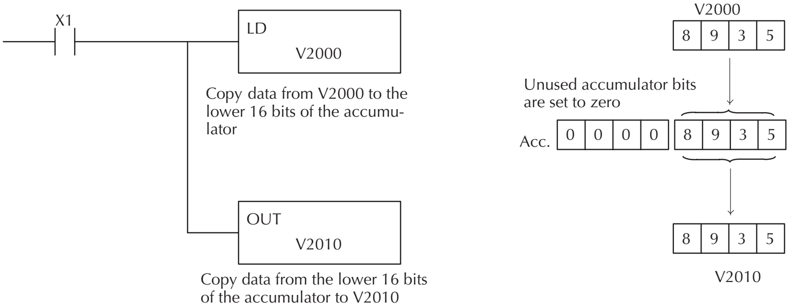
Accumulator/Stack Load and Output Data Instructions

Using the Accumulator

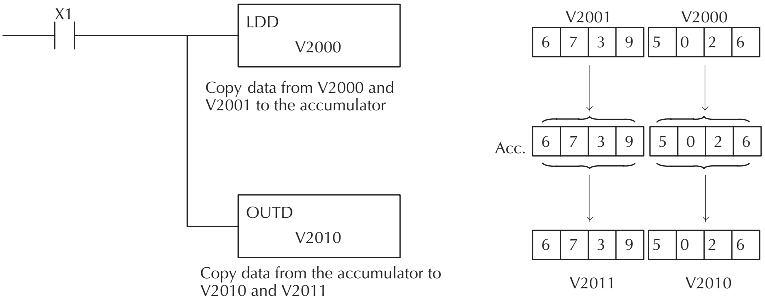
The accumulator in the D4-454 CPUs is a 32-bit register which is used as a temporary storage location for data that is being copied or manipulated in some manner. For example, you have to use the accumulator to perform math operations such as add, subtract, multiply, etc. Since there are 32 bits, you can use up to an 8-digit BCD number. 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 V2000 to V-memory location V2010.

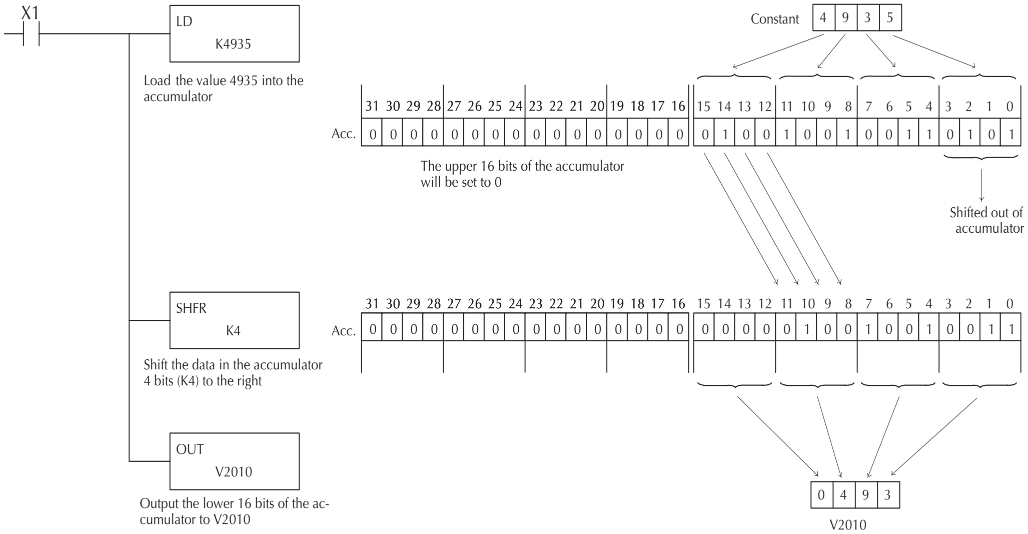

X1
LD
V2000
Copy data from V2000 to the lower 16 bits of the accumulator
OUT
V2010
Copy data from the lower 16 bits of the accumulator to V2010
V2000
8 9 3 5
Unused accumulator bits are set to zero
Acc.
0 0 0 0 8 9 3 5
8 9 3 5
V2010

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 the accumulator to V-memory. For example, if you wanted to copy data from V2000 and V2001 to V2010 and V2011 the most efficient way to perform this function would be as follows:


X1
LDD
V2000
Copy data from V2000 and V2001 to the accumulator
OUTD
V2010
Copy data from the accumulator to V2010 and V2011
V2001
V2000
6 7 3 9 5 0 2 6
Acc.
6 7 3 9 5 0 2 6
6 7 3 9 5 0 2 6
V2011
V2010

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 V2010.



Load the value 4935 into the accumulator

The upper 16 bits of the accumulator will be set to 0

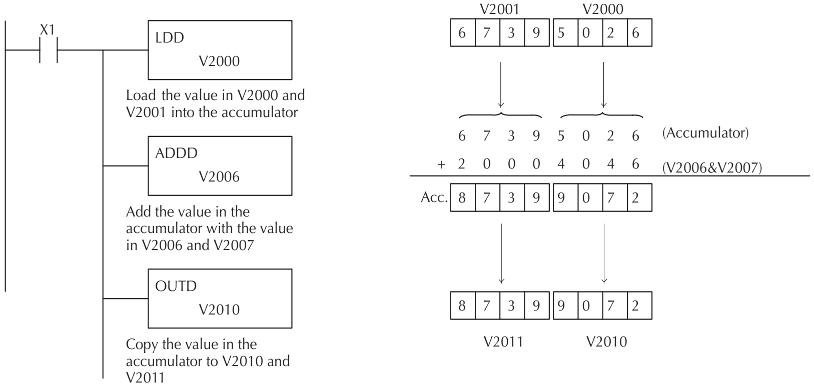
Shifted out of accumulator

Shift the data in the accumulator 4 bits (K4) to the right

Output the lower 16 bits of the accumulator to V2010

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.

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.



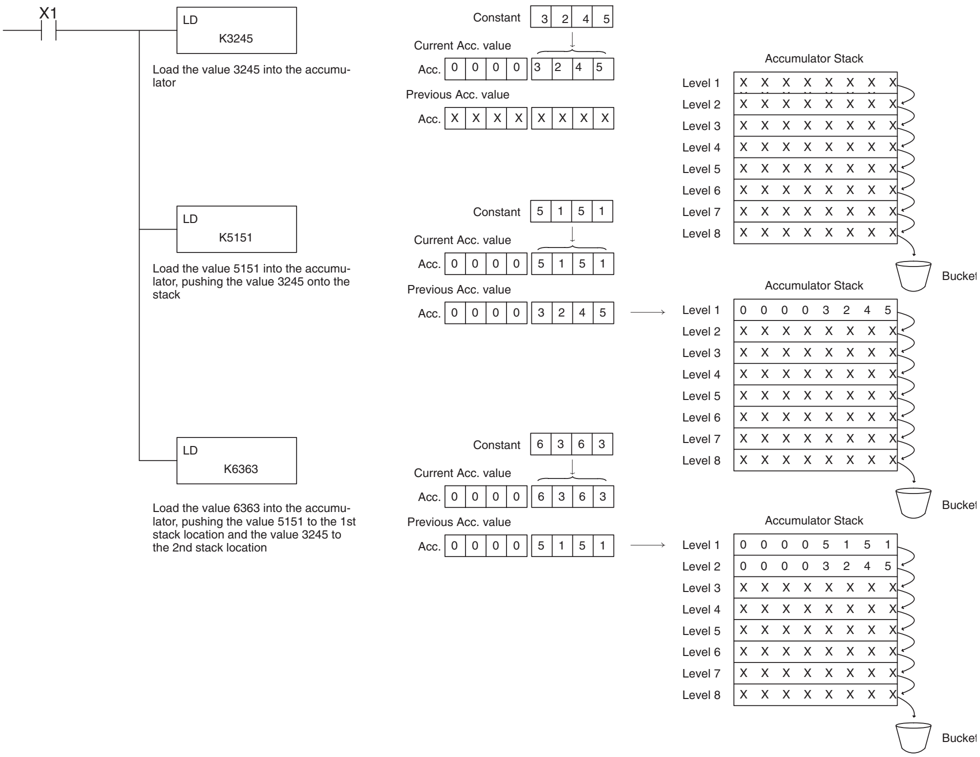
Load the value in V2000 and V2001 into the accumulator

Add the value in the accumulator with the value in V2006 and V2007

Copy the value in the accumulator to V2010 and V2011

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.

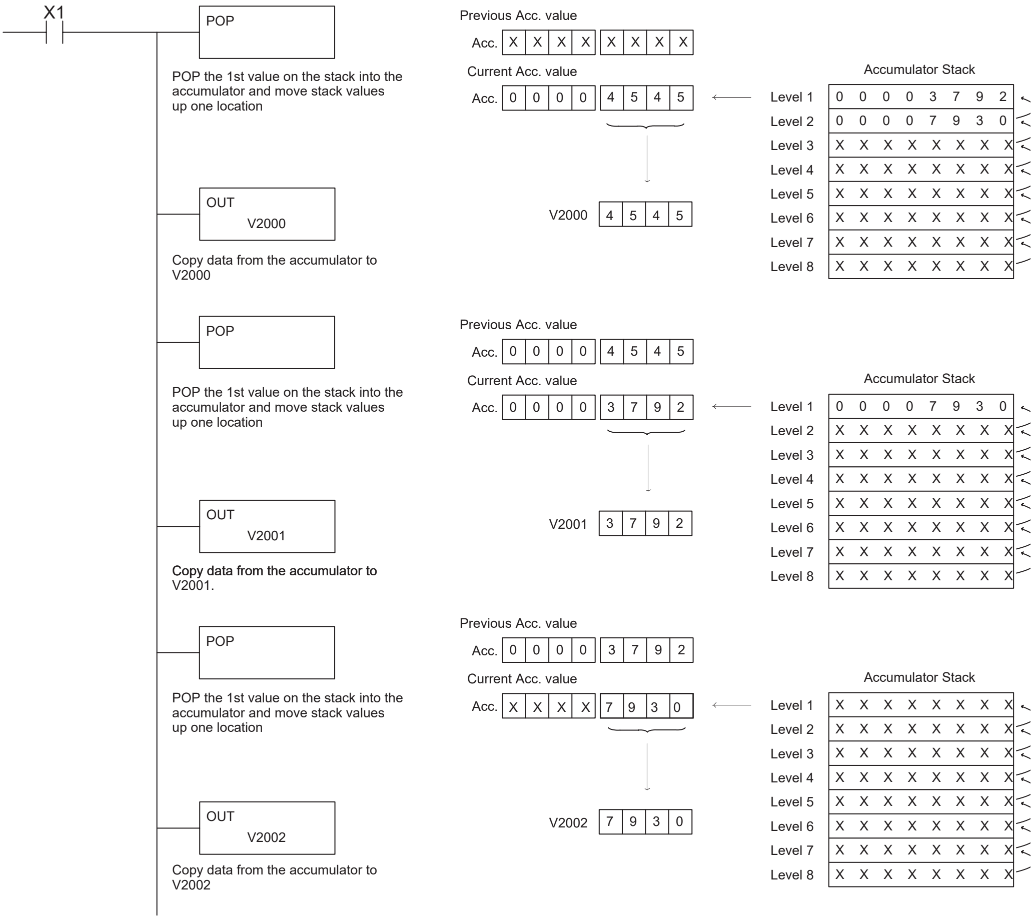


Load the value 3245 into the accumulator

Load the value 5151 into the accumulator, pushing the value 3245 onto the stack

Load the value 6363 into the accumulator, pushing the value 5151 to the 1st stack location and the value 3245 to the 2nd stack location

The POP instruction rotates values upward through the stack into the accumulator. When a POP is executed, the value which 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.



POP the 1st value on the stack into the accumulator and move stack values up one location

Copy data from the accumulator to V2000

POP the 1st value on the stack into the accumulator and move stack values up one location

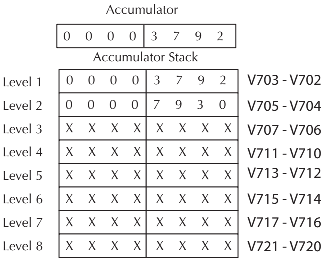
Copy data from the accumulator to V2001.

POP the 1st value on the stack into the accumulator and move stack values up one location

Copy data from the accumulator to V2002

Accumulator and Accumulator Stack Memory Locations

There may be times when you want to read a value that been placed onto the accumulator stack without having to pop the stack first. Both the accumulator and the accumulator stack have corresponding V-memory locations that can be accessed by the program. You cannot write to these locations, but you can read them or use them in comparative boolean instructions, etc.


Accumulator

0 0 0 0 3 7 9 2

Accumulator Stack

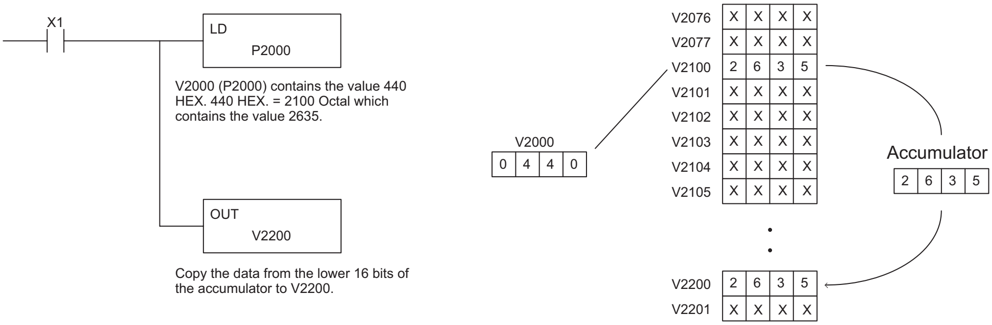
Level 1	0 0 0 0 3 7 9 2	V703 - V702
Level 2	0 0 0 0 7 9 3 0	V705 - V704
Level 3	X X X X X X X X	V707 - V706
Level 4	X X X X X X X X	V711 - V710
Level 5	X X X X X X X X	V713 - V712
Level 6	X X X X X X X X	V715 - V714
Level 7	X X X X X X X X	V717 - V716
Level 8	X X X X X X X X	V721 - V720

Using Pointers

Many of the D4-454 series instructions will allow V-memory pointers as operands commonly known as indirect addressing. Pointers allow instructions to obtain data from V-memory locations referenced by the pointer value.

NOTE: *D4-454 V-memory addressing is in octal. However, the pointers reference a V-memory location with values viewed as HEX. Use the Load Address (LDA) instruction to move an address into the pointer location. This instruction performs the Octal to Hexadecimal conversion automatically.*

In the following example we are using a pointer operand in a Load instruction. V-memory location 2000 is being used as the pointer location. V2000 contains the value 440 which the CPU views as the Hex equivalent of the Octal address V-memory location V2100. The CPU will copy the data from V2100, which (in this example) contains the value 2635, into the lower word of the accumulator.

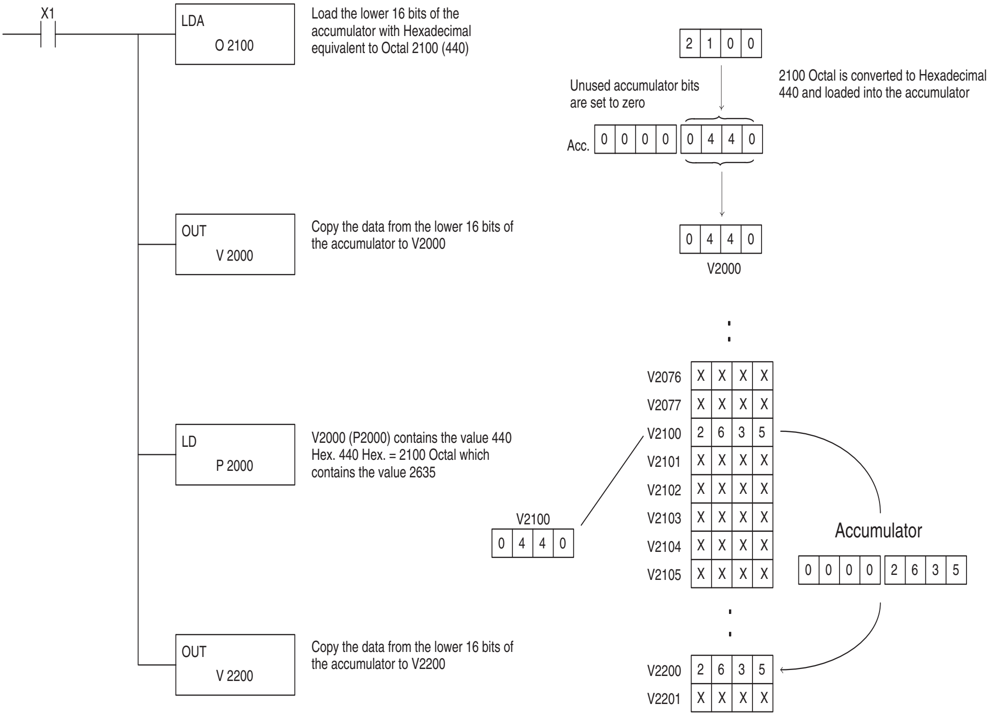


V2000 (P2000) contains the value 440 HEX. 440 HEX. = 2100 Octal which contains the value 2635.

Copy the data from the lower 16 bits of the accumulator to V2200.

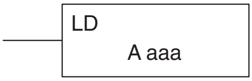
V2076	X	X	X	X
V2077	X	X	X	X
V2100	2	6	3	5
V2101	X	X	X	X
V2102	X	X	X	X
V2103	X	X	X	X
V2104	X	X	X	X
V2105	X	X	X	X
V2200	2	6	3	5
V2201	X	X	X	X

The following example is identical to the one on the previous page, with one exception. The LDA (Load Address) instruction automatically converts the Octal address to Hex.


X1
LDA
O 2100
Load the lower 16 bits of the accumulator with Hexadecimal equivalent to Octal 2100 (440)
2 1 0 0
Unused accumulator bits are set to zero
2100 Octal is converted to Hexadecimal 440 and loaded into the accumulator
Acc. 0 0 0 0 0 4 4 0
OUT
V 2000
Copy the data from the lower 16 bits of the accumulator to V2000
0 4 4 0
V2000
V2076 X X X X
V2077 X X X X
V2100 2 6 3 5
V2101 X X X X
V2102 X X X X
V2103 X X X X
V2104 X X X X
V2105 X X X X
LD
P 2000
V2000 (P2000) contains the value 440 Hex. 440 Hex. = 2100 Octal which contains the value 2635
V2100
0 4 4 0
Accumulator
0 0 0 0 2 6 3 5
OUT
V 2200
Copy the data from the lower 16 bits of the accumulator to V2200
V2200 2 6 3 5
V2201 X X X X

Load (LD)

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.

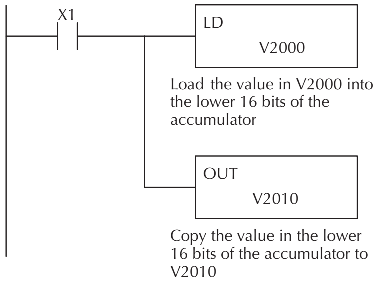


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0 – FFFF

Discrete Bit Flags	Description
SP70	On anytime the value in the accumulator is negative.
SP76	On when any instruction loads a value of zero into the accumulator.

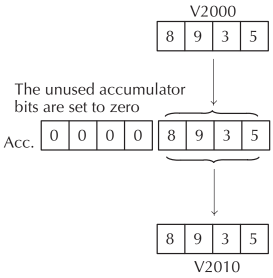
***NOTE:** Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.*

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator and output to V2010.



Load the value in V2000 into the lower 16 bits of the accumulator

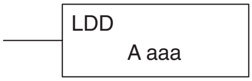
Copy the value in the lower 16 bits of the accumulator to V2010



The unused accumulator bits are set to zero

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.

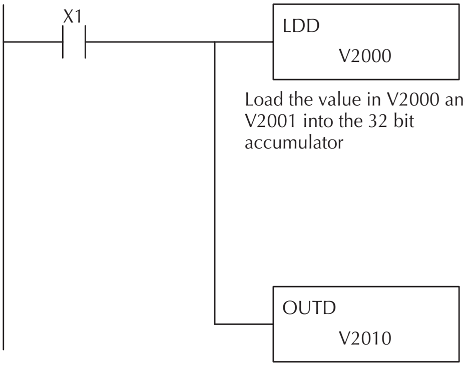


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0 – FFFFFFFF

Discrete Bit Flags	Description
SP70	On anytime the value in the accumulator is negative.
SP76	On when any instruction loads a value of zero into the accumulator.

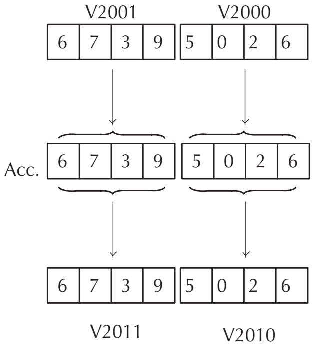
NOTE: *Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.*

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.



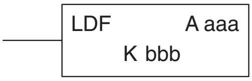
Load the value in V2000 and V2001 into the 32 bit accumulator

Copy the value in the 32 bit accumulator to V2010 and V2011



Load Formatted (LDF)

The Load Formatted instruction loads 1–32 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.

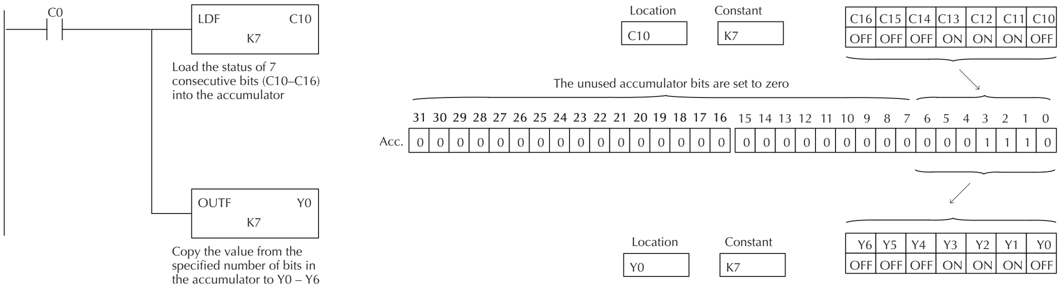


Operand Data Type		D4-454 Range	
	A	aaa	bbb
Inputs	X	0 – 1777	– –
Outputs	Y	0 – 1777	– –
Control Relays	C	0 – 3777	– –
Stage Bits	S	0 – 1777	– –
Timer Bits	T	0 – 377	– –
Counter Bits	CT	0 – 377	– –
Special Relays	SP	0 – 777	– –
Global I/O	GX	0 – 3777	
Constant	K	– –	1 – 32

Discrete Bit Flags	Description
SP70	On anytime the value in the accumulator is negative.
SP76	On when any instruction loads a value of zero into the accumulator.

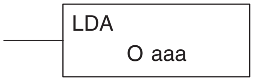
NOTE: *Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.*

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.


C0
LDF C10
K7
Load the status of 7 consecutive bits (C10–C16) into the accumulator
OUTF Y0
K7
Copy the value from the specified number of bits in the accumulator to Y0 – Y6
Location
C10
Constant
K7
C16 C15 C14 C13 C12 C11 C10
OFF OFF OFF ON ON ON OFF
The unused accumulator bits are set to zero
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0
Location
Y0
Constant
K7
Y6 Y5 Y4 Y3 Y2 Y1 Y0
OFF OFF OFF ON ON ON OFF

Load Address (LDA)

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 D4-454 system are in octal.

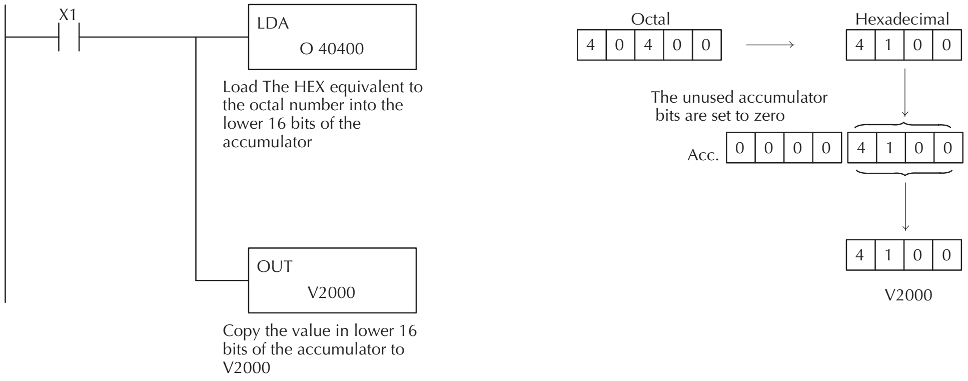


Operand Data Type		D4-454 Range
		aaa
Octal Address	O	1400 – 7377 10000 – 36777

Discrete Bit Flags	Description
SP70	On anytime the value in the accumulator is negative.
SP76	On when any instruction loads a value of zero into the accumulator.

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.



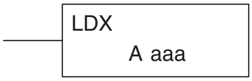
Load The HEX equivalent to the octal number into the lower 16 bits of the accumulator

Copy the value in lower 16 bits of the accumulator to V2000

The unused accumulator bits are set to zero

Load Accumulator Indexed (LDX)

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.



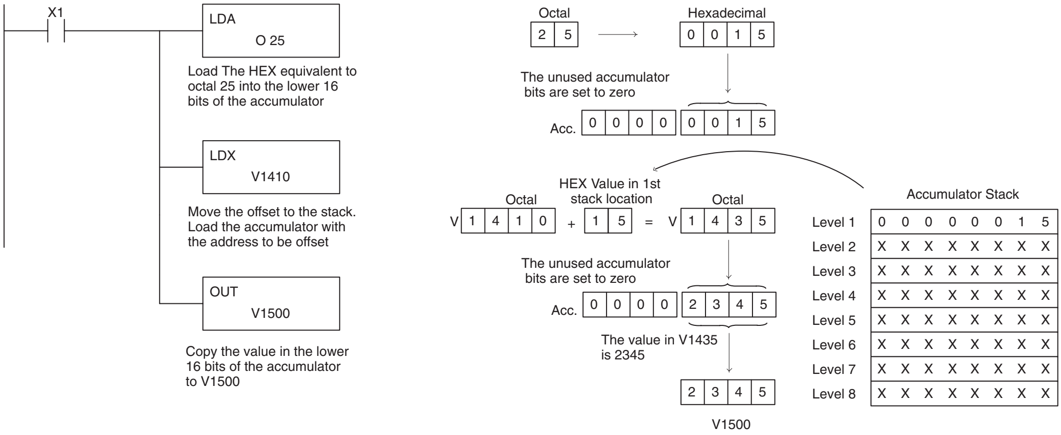
Helpful Hint: The Load Address instruction can be used to convert an octal address to a HEX address and load the value into the accumulator.

Operand Data Type		D4-454 Range	
A		aaa	aaa
V-memory	V	See memory map	See memory map
Pointer	P	See memory map	See memory map

Discrete Bit Flags	Description
SP70	On anytime the value in the accumulator is negative.
SP76	On when any instruction loads a value of zero into the accumulator.

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.


X1
LDA
O 25
Load The HEX equivalent to octal 25 into the lower 16 bits of the accumulator
LDX
V1410
Move the offset to the stack. Load the accumulator with the address to be offset
OUT
V1500
Copy the value in the lower 16 bits of the accumulator to V1500
Octal
2 5
Hexadecimal
0 0 1 5
The unused accumulator bits are set to zero
Acc.
0 0 0 0 0 0 1 5
HEX Value in 1st stack location
Octal
V 1 4 1 0
+
1 5
=
Octal
V 1 4 3 5
The unused accumulator bits are set to zero
Acc.
0 0 0 0 2 3 4 5
The value in V1435 is 2345
2 3 4 5
V1500
Accumulator Stack

Level 1	0	0	0	0	0	0	1	5
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

Load Accumulator Indexed from Data Constants (LDSX)

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.

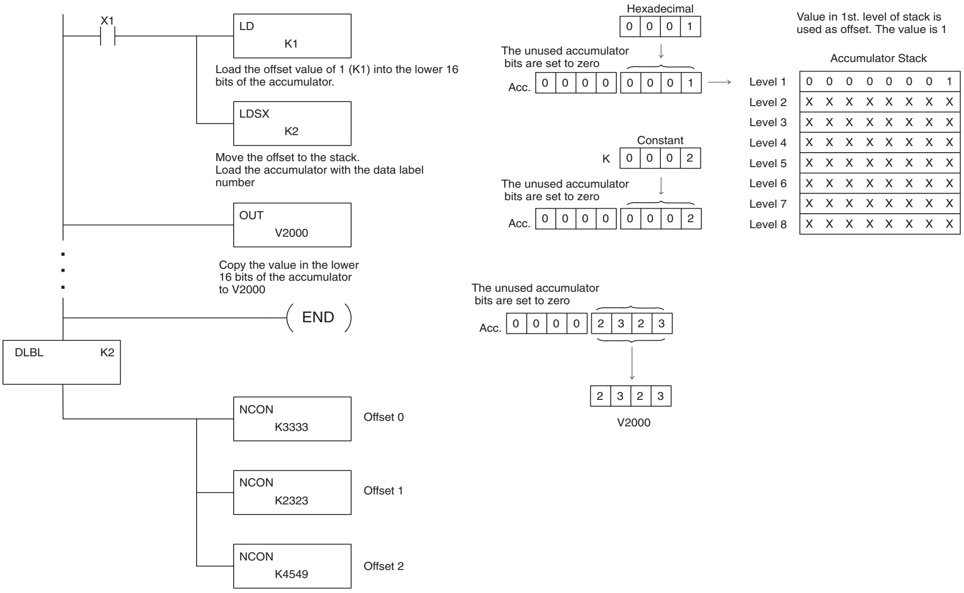
The LDSX instruction uses the value in the first level of the accumulator stack as an offset to determine which numerical or ASCII constant within the Data Label Area will be loaded into the accumulator. The LDSX instruction interprets the value in the first level of the accumulator stack as a HEX value.

Helpful Hint: The Load Address instruction can be used to convert octal to HEX and load the value into the accumulator.

Operand Data Type		D4-454 Range
		aaa
Constant	K	1-FFFF

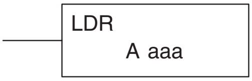
***NOTE:** Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.*

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.


X1
LD
K1
Load the offset value of 1 (K1) into the lower 16 bits of the accumulator.
LDSX
K2
Move the offset to the stack.
Load the accumulator with the data label number
OUT
V2000
Copy the value in the lower 16 bits of the accumulator to V2000
END
DLBL K2
NCON
K3333
Offset 0
NCON
K2323
Offset 1
NCON
K4549
Offset 2
Hexadecimal
0 0 0 1
The unused accumulator bits are set to zero
Acc. 0 0 0 0 0 0 0 1
Constant
K 0 0 0 2
The unused accumulator bits are set to zero
Acc. 0 0 0 0 0 0 0 2
The unused accumulator bits are set to zero
Acc. 0 0 0 0 2 3 2 3
2 3 2 3
V2000
Value in 1st. level of stack is used as offset. The value is 1
Accumulator Stack
Level 1 0 0 0 0 0 0 0 1
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

Load Real Number (LDR)

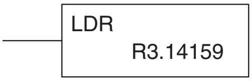
The Load Real Number instruction loads a real number contained in two consecutive V-memory locations, or an 8-digit constant into the accumulator.



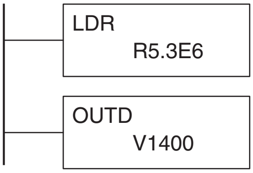
Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Real Constant	R	-3.402823E^{+38} to + +3.402823E^{+38}

Discrete Bit Flags	Description
SP70	On anytime the value in the accumulator is negative.
SP76	On when any instruction loads a value of zero into the accumulator.

DirectSOFT 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".

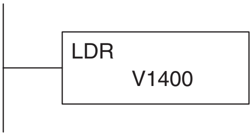


For very large numbers or very small numbers, you can use exponential notation. The number to the right is 5.3 million. The OUTD instruction stores it in V1400 and V1401.



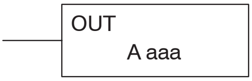
These real numbers are in the IEEE 32-bit floating point format, so they occupy two V-memory locations, regardless of how big or small the number may be! If you view a stored real number in hex, binary, or even BCD, the number shown will be very difficult to decipher. Just like all other number types, 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.



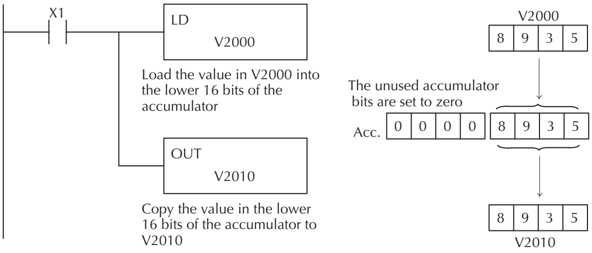
Out (OUT)

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).



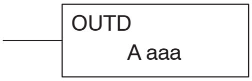
Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map

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.


X1
LD
V2000
Load the value in V2000 into the lower 16 bits of the accumulator
OUT
V2010
Copy the value in the lower 16 bits of the accumulator to V2010
V2000
8 9 3 5
The unused accumulator bits are set to zero
Acc.
0 0 0 0 8 9 3 5
8 9 3 5
V2010

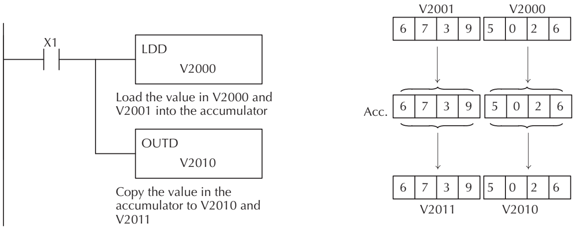
Out Double (OUTD)

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).



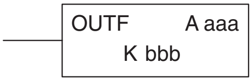
Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map

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 OUTD instruction.


X1
LDD
V2000
Load the value in V2000 and V2001 into the accumulator
OUTD
V2010
Copy the value in the accumulator to V2010 and V2011
V2001
V2000
6 7 3 9 5 0 2 6
Acc.
6 7 3 9 5 0 2 6
6 7 3 9 5 0 2 6
V2011
V2010

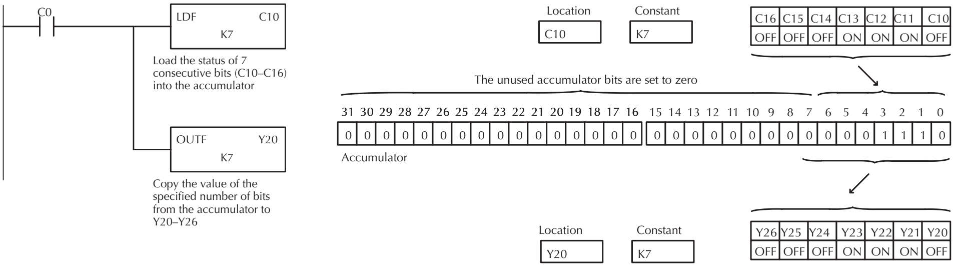
Out Formatted (OUTF)

The Out Formatted instruction outputs 1 – 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.



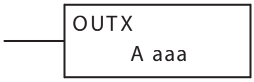
Operand Data Type		D4-454 Range	
	A	aaa	bbb
Inputs	X	0 – 1777	– –
Outputs	Y	0 – 1777	– –
Control Relays	C	0 – 3777	– –
Constant	K	– –	1 – 32

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 OUTF instruction.



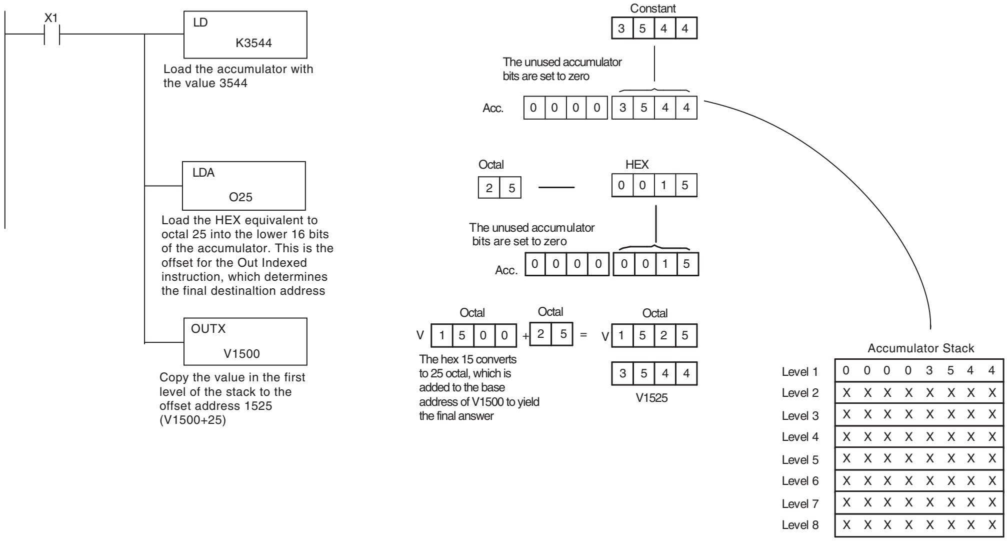
Out Indexed (OUTX)

The OUTX 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.



Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	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 LDA instruction is executed. Remember, two consecutive LD instructions places the value of the first load instruction onto the stack. The LDA instruction converts octal 25 to HEX 15 and places the value in the accumulator. The OUTX instruction outputs the value 3544 which resides in the first level of the accumulator stack to V1525.


X1
LD
K3544
LDA
O25
OUTX
V1500
Constant
3 5 4 4
The unused accumulator bits are set to zero
Acc. 0 0 0 0 3 5 4 4
Octal
2 5
HEX
0 0 1 5
The unused accumulator bits are set to zero
Acc. 0 0 0 0 0 0 1 5
Octal
V 1 5 0 0
Octal
+ 2 5
=
Octal
V 1 5 2 5
3 5 4 4
V1525
The hex 15 converts to 25 octal, which is added to the base address of V1500 to yield the final answer
Accumulator Stack

Load the accumulator with the value 3544

Load the HEX equivalent to octal 25 into the lower 16 bits of the accumulator. This is the offset for the Out Indexed instruction, which determines the final destinaltion address

Copy the value in the first level of the stack to the offset address 1525 (V1500+25)

Level 1	0	0	0	0	3	5	4	4
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

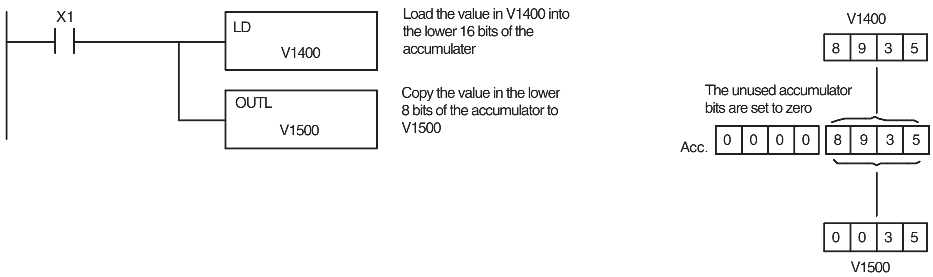
Out Least (OUTL)

The OUTL 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).

```
──┤ OUTL
      A aaa
```

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	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 LD instruction. The value in the lower 8 bits of the accumulator is copied to V1500 using the OUTL instruction.


X1
LD
V1400
OUTL
V1500
Load the value in V1400 into the lower 16 bits of the accumulater
Copy the value in the lower 8 bits of the accumulator to V1500
V1400
8 9 3 5
The unused accumulator bits are set to zero
Acc.
0 0 0 0 8 9 3 5
0 0 3 5
V1500

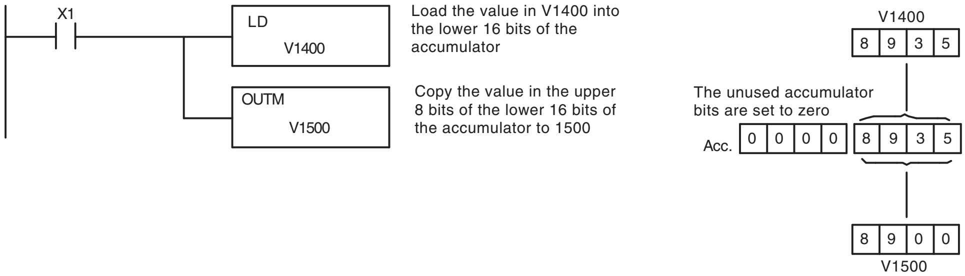
Out Most (OUTM)

The OUTM instruction copies the value in the upper eight bits of the lower sixteen bits of the accumulator to the upper eight bits of the specified V-memory location (i.e., it copies the high byte of the low word of the accumulator).

```
──┤ OUTM
      A aaa
```

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	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 LD instruction. The value in the upper 8 bits of the lower 16 bits of the accumulator is copied to V1500 using the OUTM instruction.

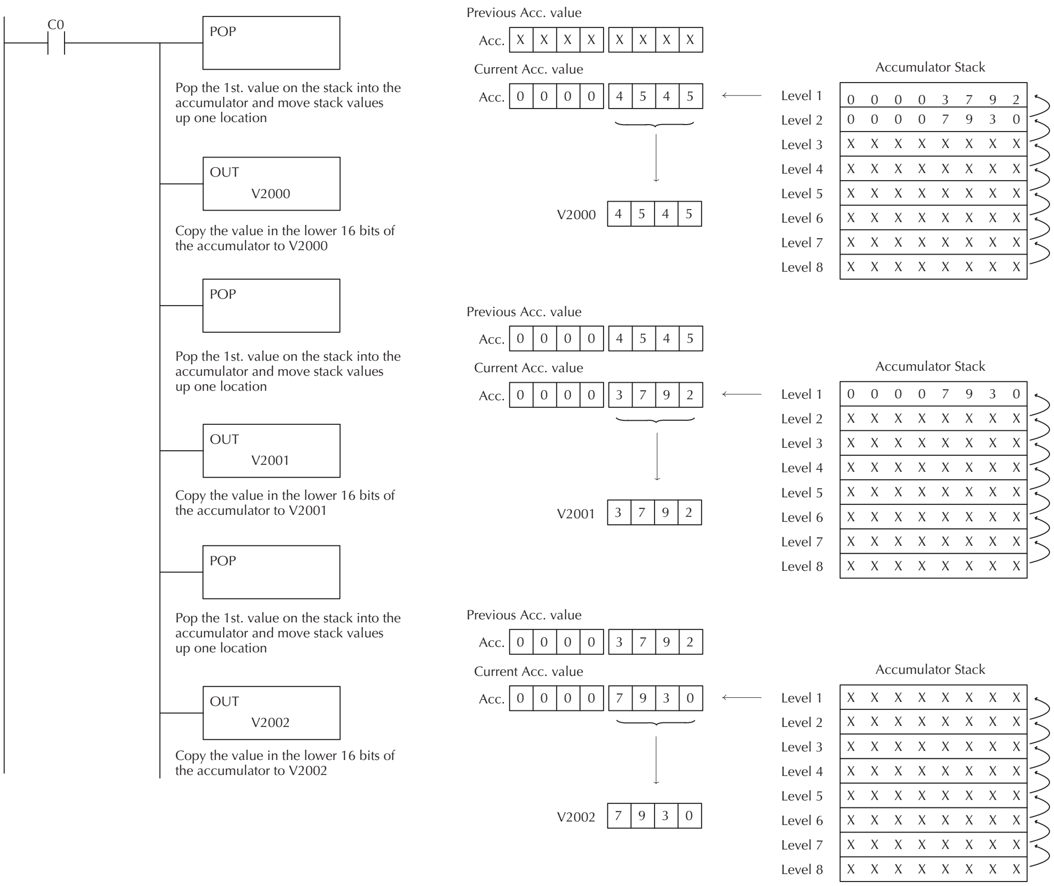

X1
LD
V1400
OUTM
V1500
Load the value in V1400 into the lower 16 bits of the accumulator
Copy the value in the upper 8 bits of the lower 16 bits of the accumulator to 1500
V1400
8 9 3 5
The unused accumulator bits are set to zero
Acc.
0 0 0 0 8 9 3 5
8 9 0 0
V1500

Pop (POP)

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.

Discrete Bit Flags	Description
SP63	ON when the result of the instruction causes the value in the accumulator to be zero.

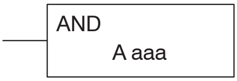
In the example below, when C0 is on, the value 4545 that was on top of the stack is moved into the accumulator using the Pop instruction The value is output to V2000 using the OUT instruction. The next Pop moves the value 3792 into the accumulator and outputs the value to V2001. The last Pop moves the 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 OUTD instruction would be used and 2 V-memory locations for each OUTD must be allocated.



Logical Instructions (Accumulator)

And (AND logical)

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.

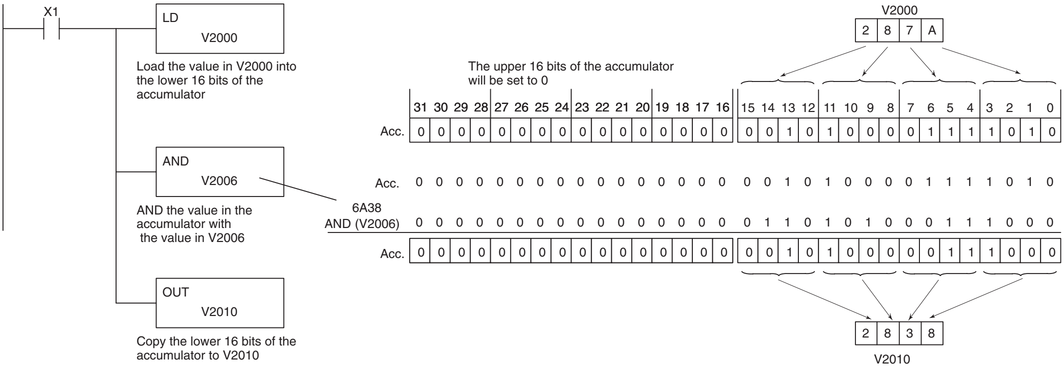


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map

Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.
SP76	ON when the value loaded into the accumulator by any instruction is zero.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the LD 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.


X1
LD
V2000
Load the value in V2000 into the lower 16 bits of the accumulator
AND
V2006
AND the value in the accumulator with the value in V2006
OUT
V2010
Copy the lower 16 bits of the accumulator to V2010
V2000
2 8 7 A
The upper 16 bits of the accumulator will be set to 0
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
6A38
AND (V2006) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0
2 8 3 8
V2010

And Double (ANDD)

ANDD 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 ANDD is zero or a negative number (the most significant bit is on).

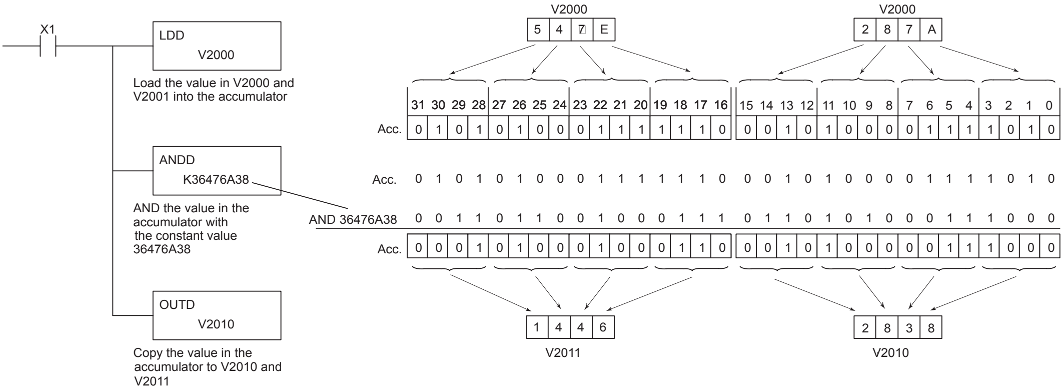
```
ANDD
   K aaa
```

Operand Data Type		D4-454 Range
		aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0 – FFFFFFFF

Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.
SP76	ON when the value loaded into the accumulator by any instruction is zero.

***NOTE:** Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the LDD instruction. The value in the accumulator is ANDed with 36476A38 using the ANDD instruction. The value in the accumulator is output to V2010 and V2011 using the OUTD instruction.



Load the value in V2000 and V2001 into the accumulator

AND the value in the accumulator with the constant value 36476A38

Copy the value in the accumulator to V2010 and V2011

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	1	0	1	0	1	0	0	0	1	1	1	1	1	1	0	0	0	1	0	1	0	0	0	0	1	1	1	1	0	1	0
Acc.	0	1	0	1	0	1	0	0	0	1	1	1	1	1	1	0	0	0	1	0	1	0	0	0	0	1	1	1	1	0	1	0
AND 36476A38	0	0	1	1	0	1	1	0	0	1	0	0	0	1	1	1	0	1	1	0	1	0	1	0	0	0	1	1	1	0	0	0
Acc.	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	1	1	1	0	0	0

And Formatted (ANDF)

ANDF
A aaa
K bbb

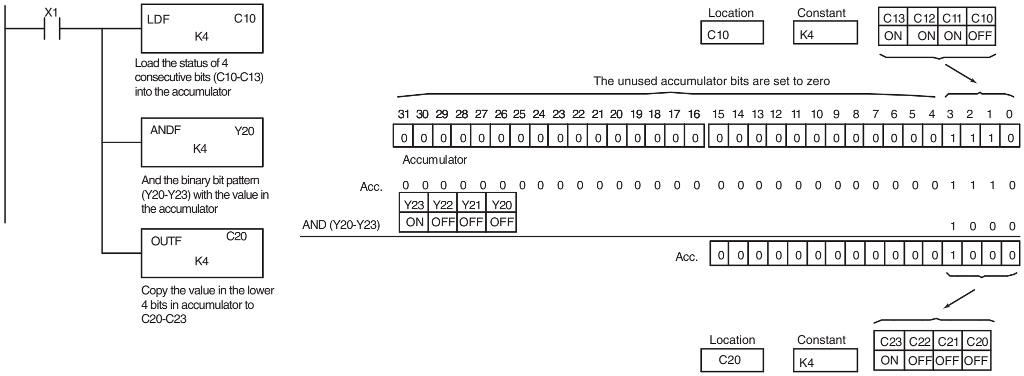
The ANDF instruction logically ANDs the binary value in the accumulator with a specified range of discrete memory bits (1 – 32). The instruction requires a starting location (Aaaa) and number of bits (Kbbb) to be ANDed. Discrete status flags indicate if the result is zero or a negative number (the most significant bit =1).

Operand Data Type		D4-454 Range	
	B	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	-
Constant	K	-	1-32

Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.
SP70	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 LDF instruction loads C10 – C13 (4 binary bits) into the accumulator. The accumulator contents is logically ANDed with the bit pattern from Y20 – Y23 using the ANDF instruction. The OUTF instruction outputs the accumulator’s lower four bits to C20 – C23.


X1
LDF C10
K4
Load the status of 4 consecutive bits (C10-C13) into the accumulator
ANDF Y20
K4
And the binary bit pattern (Y20-Y23) with the value in the accumulator
OUTF C20
K4
Copy the value in the lower 4 bits in accumulator to C20-C23
Location
C10
Constant
K4
C13 C12 C11 C10
ON ON ON OFF
The unused accumulator bits are set to zero
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0
Accumulator
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0
Y23 Y22 Y21 Y20
ON OFF OFF OFF
AND (Y20-Y23) 1 0 0 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
Location
C20
Constant
K4
C23 C22 C21 C20
ON OFF OFF OFF

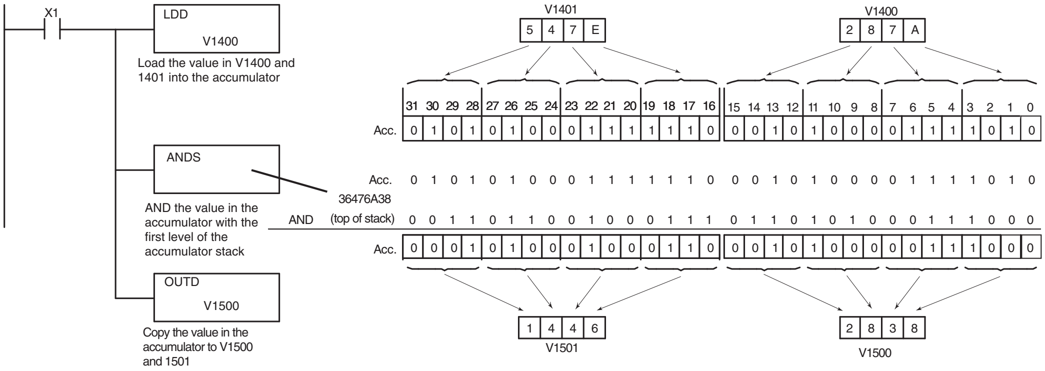
And with Stack (ANDS)

The ANDS 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 ANDS is zero or a negative number (the most significant bit is on

Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.
SP70	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 or the accumulator stack. The result resides in the accumulator. The 32-bit value is then output to V1500 and V1501.


X1
LDD
V1400
ANDS
OUTD
V1500
V1401
5 4 7 E
V1400
2 8 7 A
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc. 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
Acc. 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
36476A38
AND (top of stack) 0 0 1 1 0 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0
Acc. 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0
1 4 4 6
V1501
2 8 3 8
V1500

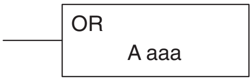
Load the value in V1400 and 1401 into the accumulator

AND the value in the accumulator with the first level of the accumulator stack

Copy the value in the accumulator to V1500 and 1501

Or (OR)

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.

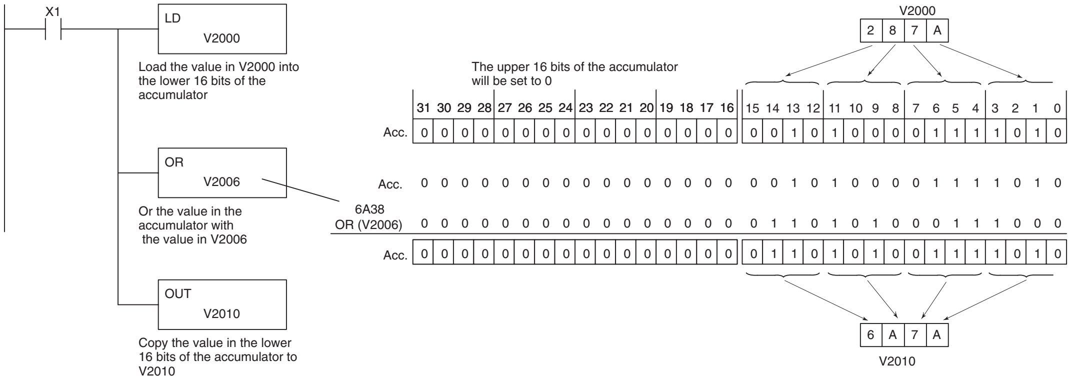


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map

Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

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 is output to V2010 using the Out instruction.


X1
LD
V2000
Load the value in V2000 into the lower 16 bits of the accumulator
OR
V2006
Or the value in the accumulator with the value in V2006
OUT
V2010
Copy the value in the lower 16 bits of the accumulator to V2010
V2000
2 8 7 A
The upper 16 bits of the accumulator will be set to 0
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
6A38
OR (V2006) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1 0 1 0
6 A 7 A
V2010

Or Double (ORD)

ORD is a 32-bit instruction that logically ORs the value in the accumulator with the value (Aaaa), which is either two consecutive V-memory locations or an 8-digit (max.) constant value. The result resides in the accumulator. Discrete status flags indicate if the result of the ORD is zero or a negative number (the most significant bit is on).

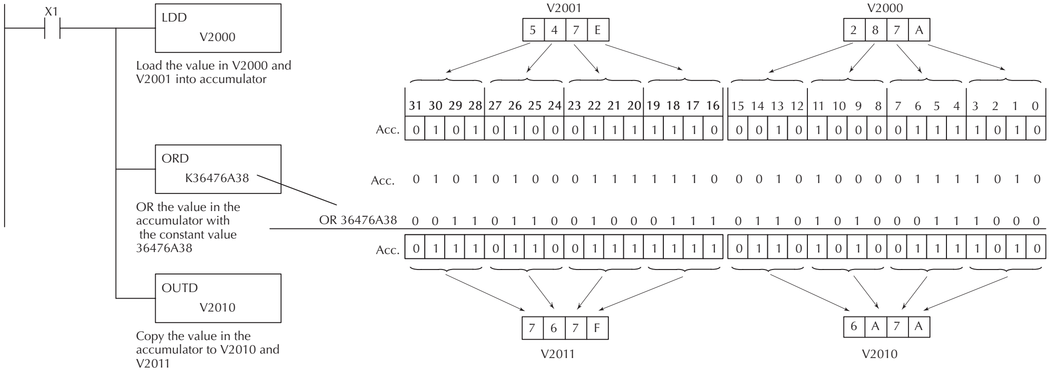
```
ORD
   K aaa
```

Operand Data Type		D4-454 Range
		aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0 – FFFFFFFF

Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.
SP70	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 value in V2000 and V2001 will be loaded into the accumulator using the LDD instruction. The value in the accumulator is OR'd with 36476A38 using the ORD instruction. The value in the accumulator is output to V2010 and V2011 using the OUTD instruction.


X1
LDD
V2000
ORD
K36476A38
OUTD
V2010
V2001
5 4 7 E
V2000
2 8 7 A
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc. 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
Acc. 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
OR 36476A38 0 0 1 1 0 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0
Acc. 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0 1 0 1 0 0 1 1 1 1 0 1 0
7 6 7 F
V2011
6 A 7 A
V2010

Load the value in V2000 and V2001 into accumulator

OR the value in the accumulator with the constant value 36476A38

Copy the value in the accumulator to V2010 and V2011

Or Formatted (ORF)

The ORF instruction logically ORs the binary value in the accumulator and a specified range of discrete bits (1 – 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).

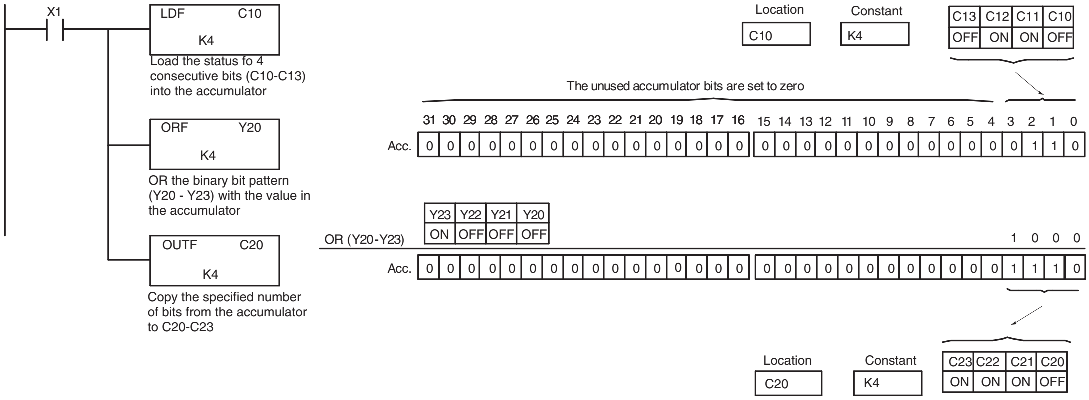
ORF A aaa
K bbb

Operand Data Type		D4-454 Range	
	A/B	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	- -
Constant	K	-	1-32

Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.
SP70	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 LDF instruction loads C10 – C13 (4 binary bits) into the accumulator. The ORF instruction logically ORs the accumulator contents with Y20 – Y23 bit pattern. The ORF instruction outputs the accumulator's lower four bits to C20 – C23.

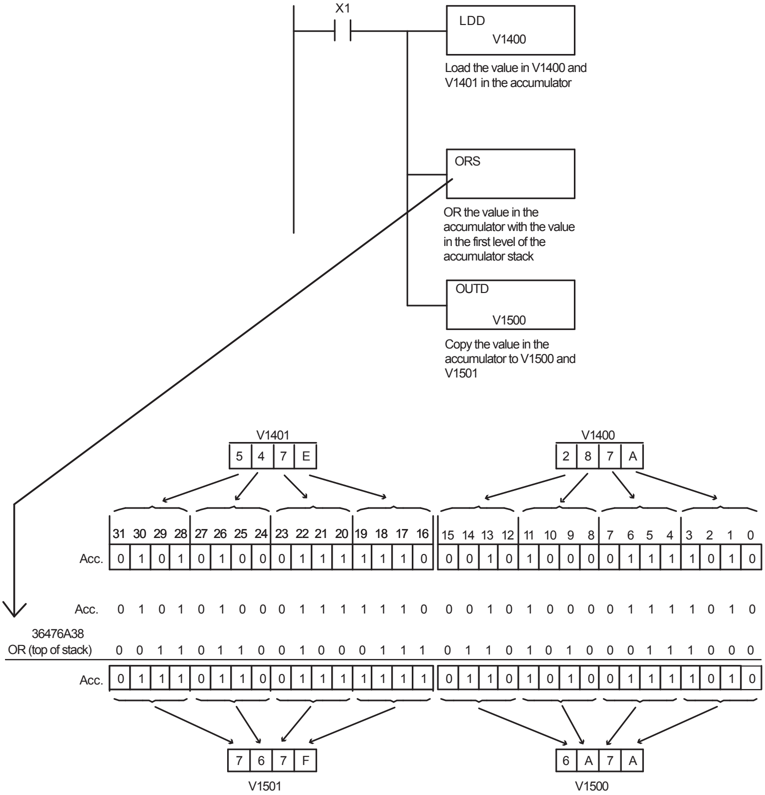

X1
LDF C10
K4
Load the status fo 4 consecutive bits (C10-C13) into the accumulator
ORF Y20
K4
OR the binary bit pattern (Y20 - Y23) with the value in the accumulator
OUTF C20
K4
Copy the specified number of bits from the accumulator to C20-C23
Location
C10
Constant
K4
C13 C12 C11 C10
OFF ON ON OFF
The unused accumulator bits are set to zero
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
Y23 Y22 Y21 Y20
ON OFF OFF OFF
OR (Y20-Y23)
1 0 0 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0
Location
C20
Constant
K4
C23 C22 C21 C20
ON ON ON OFF

Or with Stack (ORS)

The ORS 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 ORS is zero or a negative number (the most significant bit is on).

Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.
SP70	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 OR'd with the binary value in the first level of the stack. The result resides in the accumulator.



Load the value in V1400 and V1401 in the accumulator

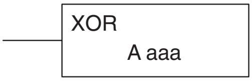
OR the value in the accumulator with the value in the first level of the accumulator stack

Copy the value in the accumulator to V1500 and V1501

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	1	0	1	0	1	0	0	0	1	1	1	1	1	1	0	0	0	1	0	1	0	0	0	0	1	1	1	1	0	1	0
Acc.	0	1	0	1	0	1	0	0	0	1	1	1	1	1	1	0	0	0	1	0	1	0	0	0	0	1	1	1	1	0	1	0
36476A38 OR (top of stack)	0	0	1	1	0	1	1	0	0	1	0	0	0	1	1	1	0	1	1	0	1	0	1	0	0	0	1	1	1	0	0	0
Acc.	0	1	1	1	0	1	1	0	0	1	1	1	1	1	1	1	0	1	1	0	1	0	1	0	0	1	1	1	1	0	1	0

Exclusive Or (XOR)

The XOR 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 in the accumulator. The discrete status flag indicates if the result of the XOR is zero.

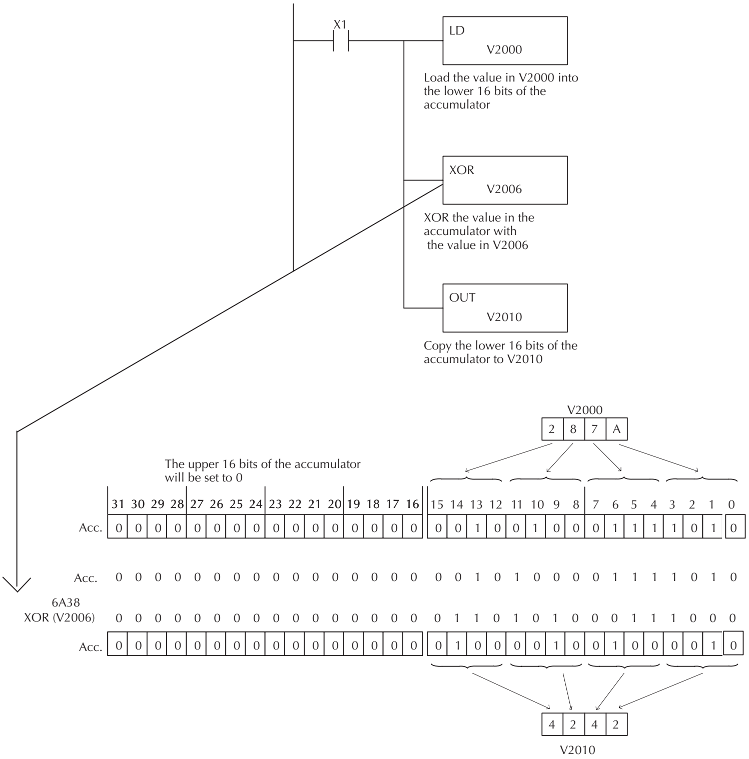


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map

Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.

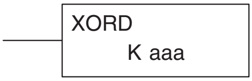
NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the LD instruction. The value in the accumulator is exclusive ORed with V2006 using the XOR instruction. The value in the lower 16 bits of the accumulator is output to V2010 using the OUT instruction.


X1
LD
V2000
Load the value in V2000 into the lower 16 bits of the accumulator
XOR
V2006
XOR the value in the accumulator with the value in V2006
OUT
V2010
Copy the lower 16 bits of the accumulator to V2010
V2000
2 8 7 A
The upper 16 bits of the accumulator will be set to 0
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 1 1 0 1 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
6A38
XOR (V2006) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0
4 2 4 2
V2010

Exclusive Or Double (XORD)

The XORD is a 32-bit instruction that performs an exclusive OR of the value in the accumulator and the value (Aaaa), which is either two consecutive V-memory locations or an 8 digit (max.) constant. The result resides in the accumulator. Discrete status flags indicate if the result of the XORD is zero or a negative number (the most significant bit is on).

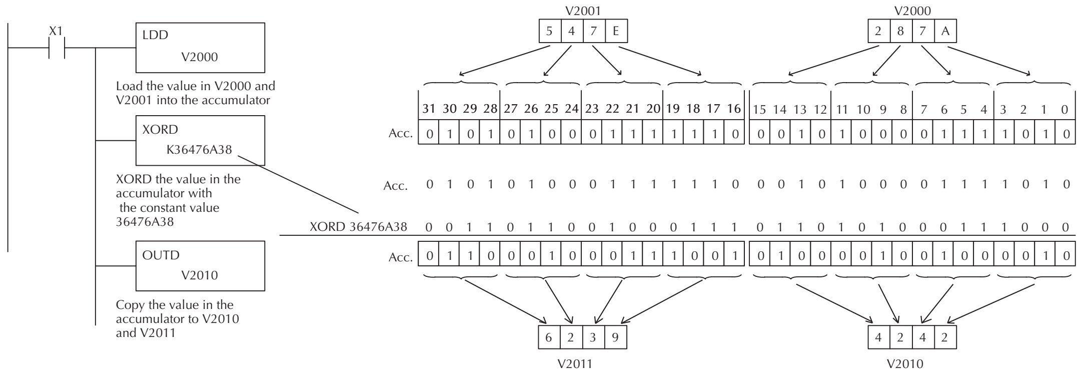


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0 – FFFFFFFF

Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.
SP70	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 value in V2000 and V2001 will be loaded into the accumulator using the LDD instruction. The value in the accumulator is exclusively ORed with 36476A38 using the XORD instruction. The value in the accumulator is output to V2010 and V2011 using the OUTD instruction.


X1
LDD
V2000
Load the value in V2000 and V2001 into the accumulator
XORD
K36476A38
XORD the value in the accumulator with the constant value 36476A38
OUTD
V2010
Copy the value in the accumulator to V2010 and V2011
V2001
5 4 7 E
V2000
2 8 7 A
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc. 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
Acc. 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
XORD 36476A38 0 0 1 1 0 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0
Acc. 0 1 1 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0
6 2 3 9
V2011
4 2 4 2
V2010

Exclusive Or Formatted (XORF)

The XORF instruction performs an exclusive OR of the binary value in the accumulator and a specified range of discrete memory bits (1–32).

```
XORF        A aaa
  K  bbb
```

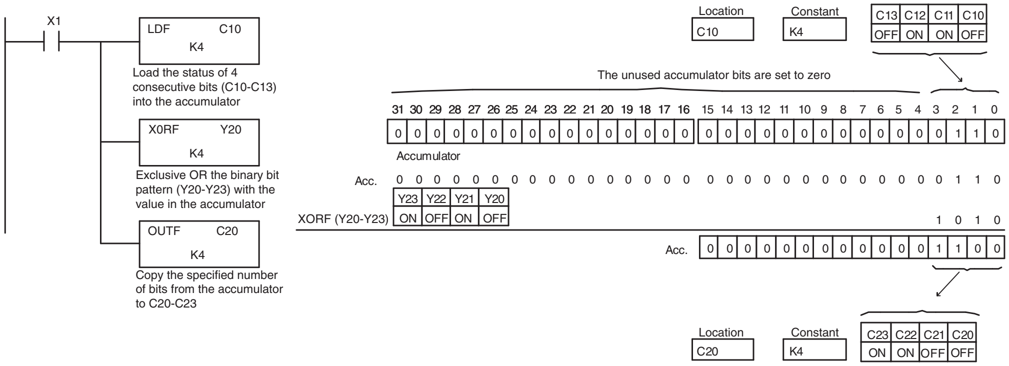
The instruction requires a starting location (Aaaa) and the number of bits (Bbbb) to be exclusive ORed. Discrete status flags indicate if the result of the XORF is zero or negative (the most significant bit =1).

Operand Data Type		D4-454 Range	
	A/B	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	0 – 3777	
Constant	K	-	1-32

Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.
SP70	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 LDF instruction. The value in the accumulator will be logically exclusive ORed with the bit pattern from Y20 – Y23 using the XORF instruction. The value in the lower 4 bits of the accumulator is output to C20 – C23 using the OUTF instruction.


X1
LDF C10
K4
Load the status of 4 consecutive bits (C10-C13) into the accumulator
XORF Y20
K4
Exclusive OR the binary bit pattern (Y20-Y23) with the value in the accumulator
OUTF C20
K4
Copy the specified number of bits from the accumulator to C20-C23
Location
C10
Constant
K4
C13 C12 C11 C10
OFF ON ON OFF
The unused accumulator bits are set to zero
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
Accumulator
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
Y23 Y22 Y21 Y20
ON OFF ON OFF
XORF (Y20-Y23)
1 0 1 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
Location
C20
Constant
K4
C23 C22 C21 C20
ON ON OFF OFF

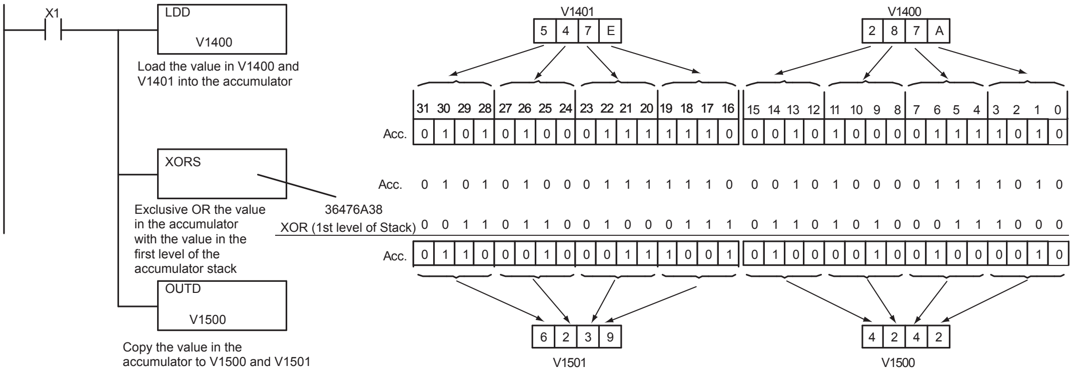
Exclusive Or with Stack (XORS)

The XORS 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 XORS is zero or a negative number (the most significant bit is on).

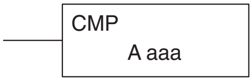
Discrete Bit Flags	Description
SP63	ON if the result in the accumulator is zero.
SP70	ON if the result in the accumulator is negative

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the LDD instruction. The binary value in the accumulator will be exclusively OR'd with 36476A38 using the XORS instruction. The value in the accumulator is output to V1500 and V1501 using the OUTD instruction.


X1
LDD
V1400
Load the value in V1400 and V1401 into the accumulator
XORS
Exclusive OR the value in the accumulator with the value in the first level of the accumulator stack
OUTD
V1500
Copy the value in the accumulator to V1500 and V1501
V1401
5 4 7 E
V1400
2 8 7 A
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc. 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
Acc. 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0
36476A38
XOR (1st level of Stack) 0 0 1 1 0 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0
Acc. 0 1 1 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0
6 2 3 9
V1501
4 2 4 2
V1500

Compare (CMP)



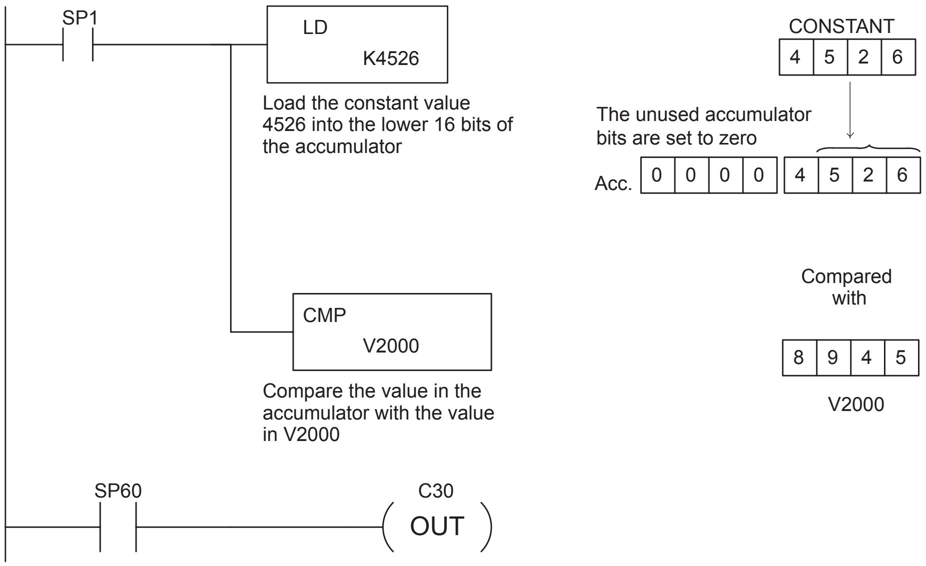
The CMP instruction is a 16-bit instruction that compares the value in the lower 16 bits of the accumulator with the value in a specified V-memory location (Aaaa). The corresponding status flag will be turned on indicating the result of the comparison. The data format for this instruction is BCD/Hex, Decimal and Binary.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	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.

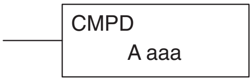
***NOTE:** Status flags are valid only until another instruction uses the same flag.*

In the following example when SP1 is on, the constant 4526 will be loaded into the lower 16 bits of the accumulator using the LD instruction. The value in the accumulator is compared with the value in V2000 using the CMP 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 CMP instruction, SP60 will turn on, energizing C30.


SP1
LD
K4526
Load the constant value 4526 into the lower 16 bits of the accumulator
CONSTANT
4 5 2 6
The unused accumulator bits are set to zero
Acc. 0 0 0 0 4 5 2 6
CMP
V2000
Compare the value in the accumulator with the value in V2000
Compared with
8 9 4 5
V2000
SP60
C30
OUT

Compare Double (CMPD)

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. The data format for this instruction is BCD/Hex, Decimal and Binary.

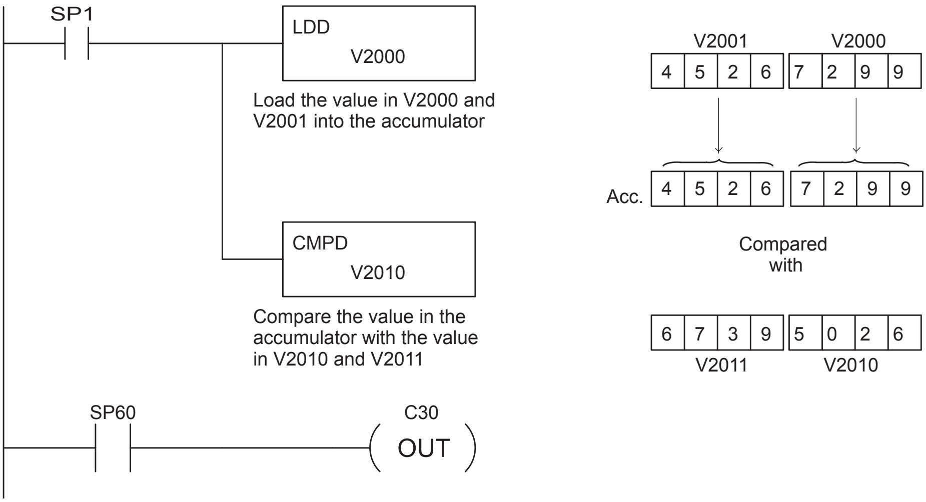


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0 – FFFFFFFF

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.

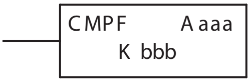
NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when SP1 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 C30.


SP1
LDD
V2000
Load the value in V2000 and V2001 into the accumulator
CMPD
V2010
Compare the value in the accumulator with the value in V2010 and V2011
SP60
C30
OUT
V2001
V2000
4 5 2 6 7 2 9 9
Acc.
4 5 2 6 7 2 9 9
Compared with
6 7 3 9 5 0 2 6
V2011
V2010

Compare Formatted (CMPF)

The Compare Formatted instruction 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. The data format for this instruction is BCD/Hex, Decimal and Binary.

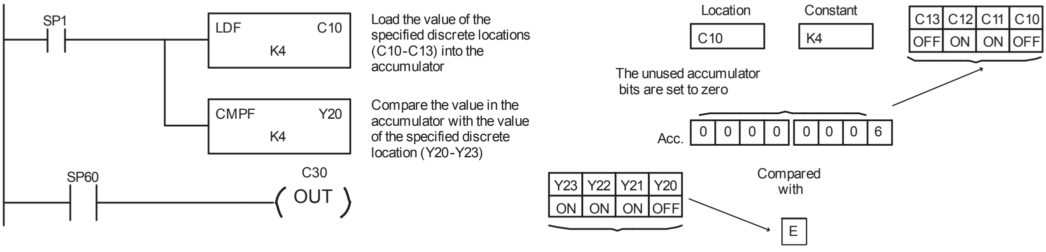


Operand Data Type		D4-454 Range	
	A/B	aaa	bbb
Inputs	X	0 – 1777	-
Outputs	Y	0 – 1777	-
Control Relays	C	0 – 3777	-
Stage Bits	S	0 – 1777	-
Timer Bits	T	0 – 377	-
Counter Bits	CT	0 – 377	-
Special Relays	SP	0 – 777	-
Global I/O	GX	0 – 3777	
Constant	K	-	1-32

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.

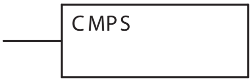
NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when SP1 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.


SP1
LDF C10
K4
Load the value of the specified discrete locations (C10-C13) into the accumulator
CMPF Y20
K4
Compare the value in the accumulator with the value of the specified discrete location (Y20-Y23)
SP60
C30
OUT
Location
C10
Constant
K4
C13 C12 C11 C10
OFF ON ON OFF
The unused accumulator bits are set to zero
Acc. 0 0 0 0 0 0 0 6
Compared with
Y23 Y22 Y21 Y20
ON ON ON OFF
E

Compare with Stack (CMPS)

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 data format for this instruction is BCD/Hex, Decimal and Binary.

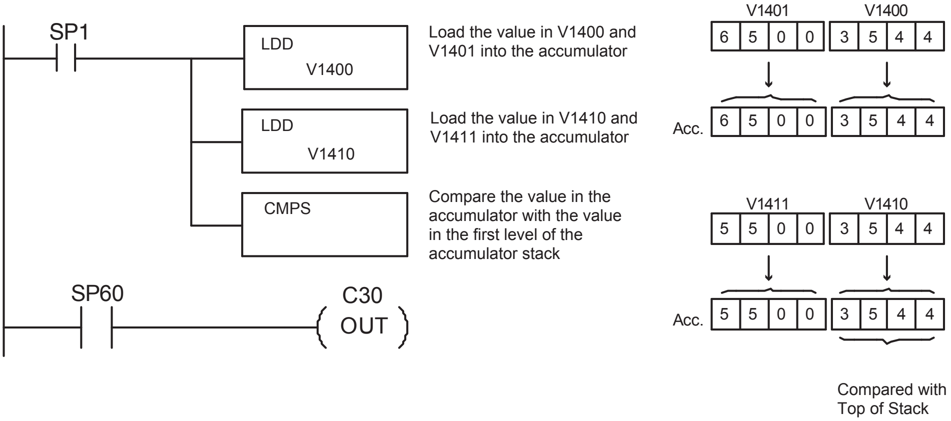


The corresponding status flag will be turned on, indicating the result of the comparison. This does not affect the value in the accumulator.

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

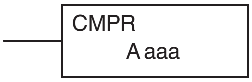
NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when SP1 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.


SP1
LDD
V1400
Load the value in V1400 and V1401 into the accumulator
LDD
V1410
Load the value in V1410 and V1411 into the accumulator
CMPS
Compare the value in the accumulator with the value in the first level of the accumulator stack
SP60
C30
OUT
V1401
V1400
6 5 0 0 3 5 4 4
Acc.
6 5 0 0 3 5 4 4
V1411
V1410
5 5 0 0 3 5 4 4
Acc.
5 5 0 0 3 5 4 4
Compared with Top of Stack

Compare Real Number (CMPR)

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.

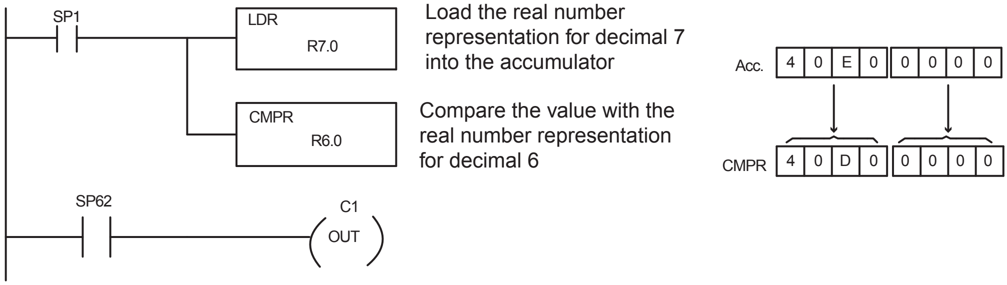


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	R	-3.402823E+38 to + +3.402823E+38

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

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when SP1 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), turning on control relay C1.

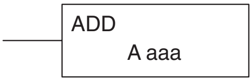


Load the real number representation for decimal 7 into the accumulator

Compare the value with the real number representation for decimal 6

Math Instructions

Add (ADD)



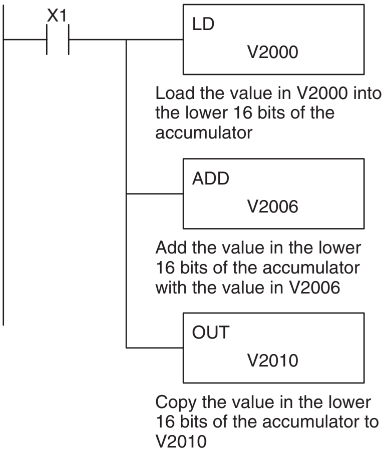
Add is a 16-bit instruction that adds a BCD value in the accumulator with a BCD value in a V-memory location (Aaaa). (You cannot use a constant as the parameter in the box.) The result resides in the accumulator

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	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 was encountered.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

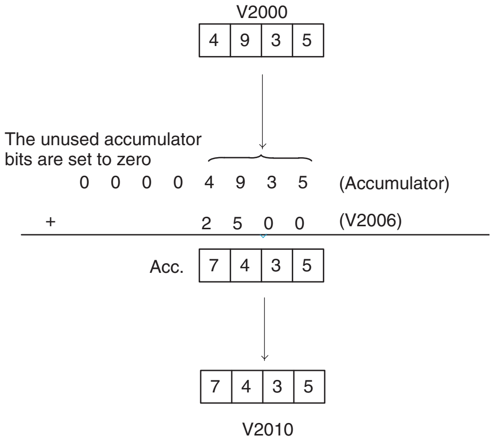
In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the lower 16 bits of the accumulator is added to the value in V2006 using the Add instruction. The value in the accumulator is copied to V2010 using the Out instruction.



Load the value in V2000 into the lower 16 bits of the accumulator

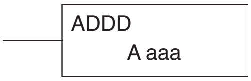
Add the value in the lower 16 bits of the accumulator with the value in V2006

Copy the value in the lower 16 bits of the accumulator to V2010


V2000
4 9 3 5
The unused accumulator bits are set to zero
0 0 0 0 4 9 3 5 (Accumulator)
+ 2 5 0 0 (V2006)
Acc. 7 4 3 5
7 4 3 5
V2010

Add Double (ADDD)

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.

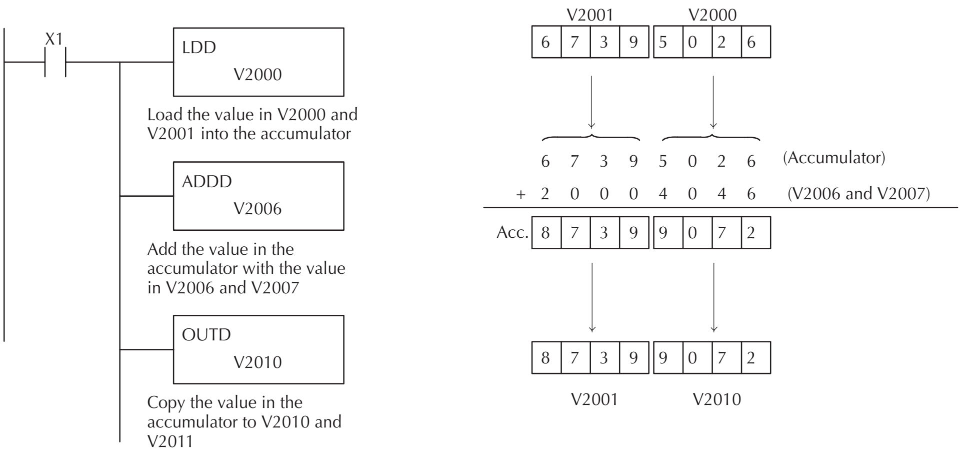


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0 – 99999999

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 was encountered.

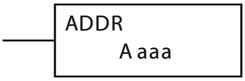
NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when X1 is on, the value in 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.


X1
LDD
V2000
Load the value in V2000 and V2001 into the accumulator
ADDD
V2006
Add the value in the accumulator with the value in V2006 and V2007
OUTD
V2010
Copy the value in the accumulator to V2010 and V2011
V2001
V2000
6 7 3 9 5 0 2 6
6 7 3 9 5 0 2 6
(Accumulator)
+ 2 0 0 0 4 0 4 6
(V2006 and V2007)
Acc.
8 7 3 9 9 0 7 2
8 7 3 9 9 0 7 2
V2001
V2010

Add Real (ADDR)

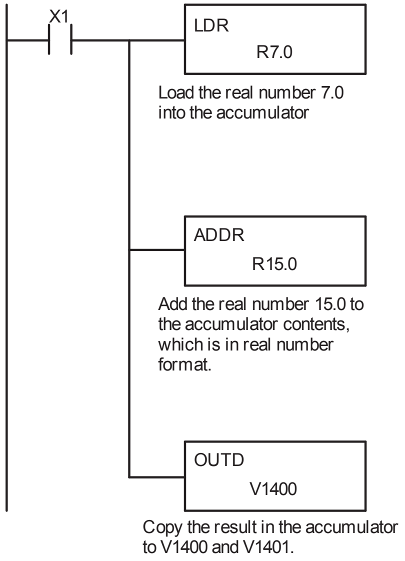
The Add Real instruction adds a real number in the accumulator with either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must be Real data type (IEEE floating point format).



Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	R	$-3.402823E^{+38}$ to + $+3.402823E^{+38}$

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.

NOTE 1: *Status flags are valid only until another instruction uses the same flag.*
NOTE 2: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use DirectSOFT programming software for this feature.



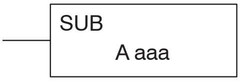
Load the real number 7.0 into the accumulator

Add the real number 15.0 to the accumulator contents, which is in real number format.

Copy the result in the accumulator to V1400 and V1401.

Subtract (SUB)

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.

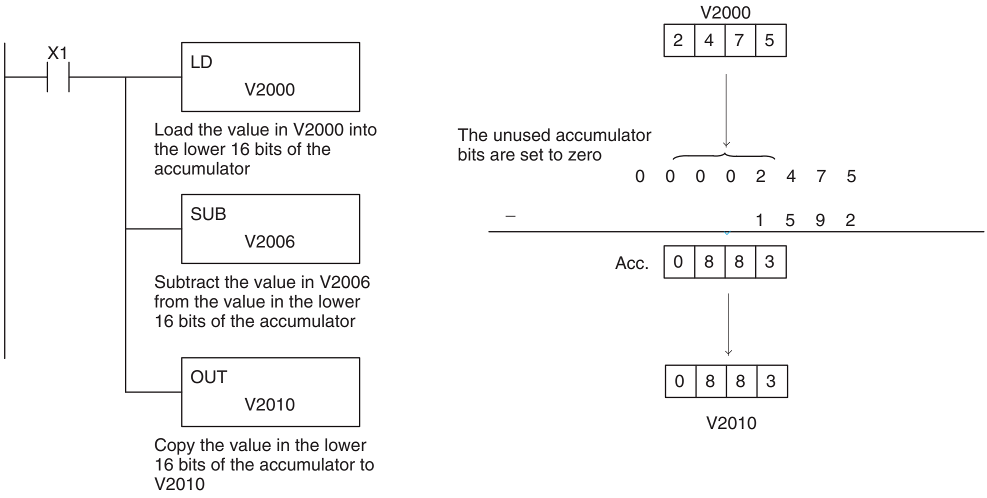


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map

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
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 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.



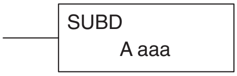
Load the value in V2000 into the lower 16 bits of the accumulator

Subtract the value in V2006 from the value in the lower 16 bits of the accumulator

Copy the value in the lower 16 bits of the accumulator to V2010

Subtract Double (SUBD)

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.

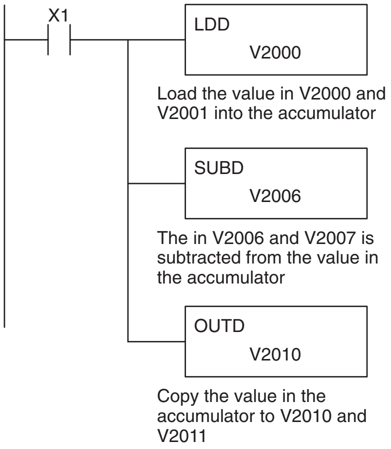


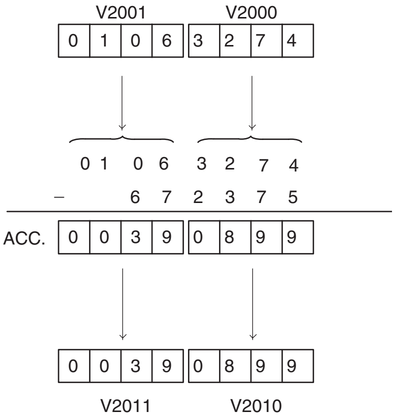
Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0 – 99999999

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
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.

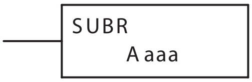
NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following example, when 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.


X1
LDD
V2000
Load the value in V2000 and V2001 into the accumulator
SUBD
V2006
The in V2006 and V2007 is subtracted from the value in the accumulator
OUTD
V2010
Copy the value in the accumulator to V2010 and V2011



Subtract Real (SUBR)

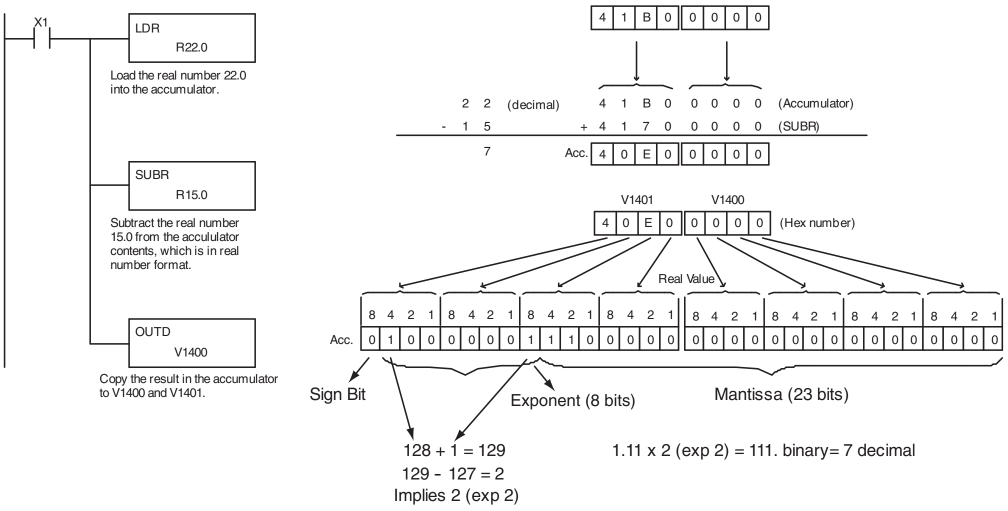


The Subtract Real is a 32-bit instruction that subtracts a real number, which is either two consecutive V-memory locations or a 32-bit constant, from a real number in the accumulator. The result is a 32-bit real number that resides in the accumulator. Both numbers must be Real data type (IEEE floating point format).

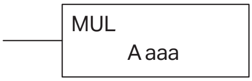
Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	R	$-3.402823E^{+38}$ to + $+3.402823E^{+38}$

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.

NOTE: *Status flags are valid only until another instruction uses the same flag.*


X1
LDR
R22.0
Load the real number 22.0 into the accumulator.
SUBR
R15.0
Subtract the real number 15.0 from the acculilator contents, which is in real number format.
OUTD
V1400
Copy the result in the accumulator to V1400 and V1401.
4 1 B 0 0 0 0 0
2 2 (decimal)
- 1 5
7
4 1 B 0 0 0 0 0 (Accumulator)
+ 4 1 7 0 0 0 0 0 (SUBR)
Acc. 4 0 E 0 0 0 0 0
V1401
V1400
4 0 E 0 0 0 0 0 (Hex number)
Real Value
8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1
Acc. 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Sign Bit
Exponent (8 bits)
Mantissa (23 bits)
128 + 1 = 129
129 - 127 = 2
Implies 2 (exp 2)
1.11 x 2 (exp 2) = 111. binary= 7 decimal

Multiply (MUL)



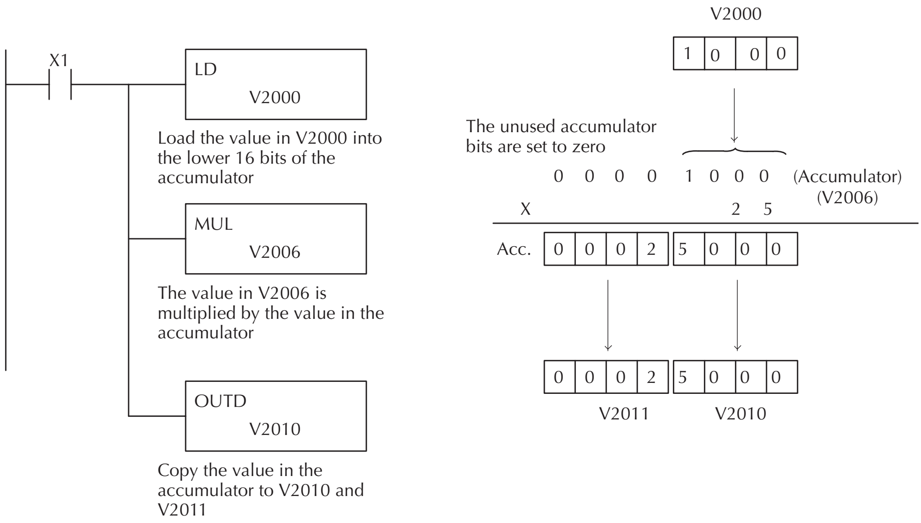
Multiply is a 16-bit instruction that multiplies the BCD value (Aaaa), which is either a V-memory location or a 4-digit (max.) constant, by the BCD value in the lower 16 bits of the accumulator. The result can be up to 8 digits and resides in the accumulator.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	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 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 V2000 will be loaded into the accumulator using the Load instruction. The value in V2006 is multiplied by the value in the accumulator using the Multiply instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



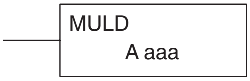
Load the value in V2000 into the lower 16 bits of the accumulator

The value in V2006 is multiplied by the value in the accumulator

Copy the value in the accumulator to V2010 and V2011

Multiply Double (MULD)

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.

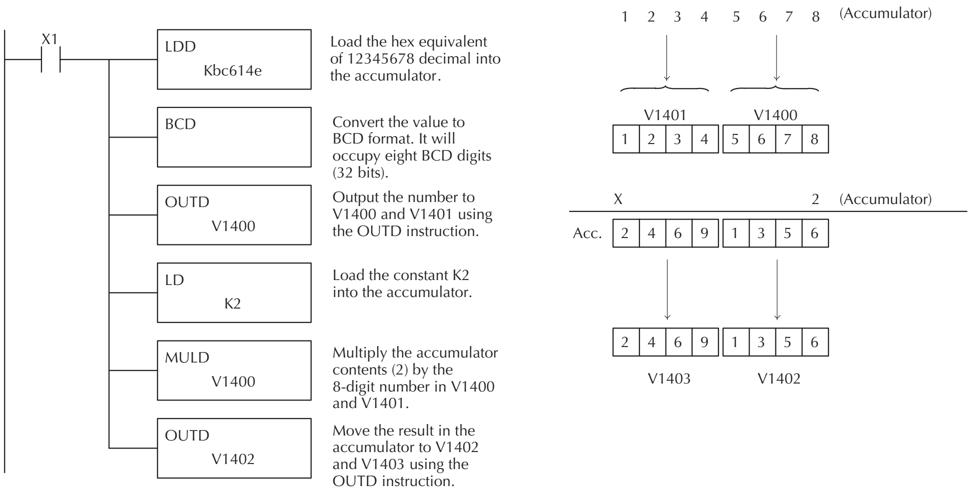


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map

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 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 gives us 24691356.



Load the hex equivalent of 12345678 decimal into the accumulator.

Convert the value to BCD format. It will occupy eight BCD digits (32 bits).

Output the number to V1400 and V1401 using the OUTD instruction.

Load the constant K2 into the accumulator.

Multiply the accumulator contents (2) by the 8-digit number in V1400 and V1401.

Move the result in the accumulator to V1402 and V1403 using the OUTD instruction.

Multiply Real (MULR)

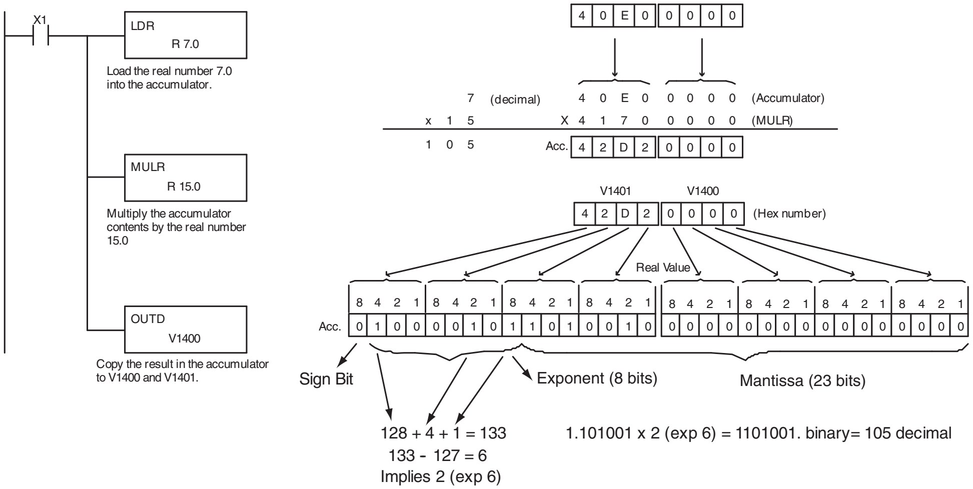
The Multiply Real instruction multiplies a real number in the accumulator with either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must be Real data type (IEEE floating point format).

```
MULR
   A aaa
```

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Real Constant	R	-3.402823E^{+38} to + +3.402823E^{+38}

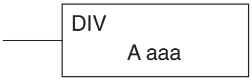
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP71	On anytime the V-memory specified by a pointer (P) is not valid.
SP72	On anytime the value in the accumulator is an invalid floating point number.
SP73	On when a signed addition or subtraction results in an incorrect sign bit.
SP74	On anytime a floating point math operation results in an underflow error.

NOTE: *Status flags are valid only until another instruction uses the same flag.*


X1
LDR
R 7.0
Load the real number 7.0 into the accumulator.
MULR
R 15.0
Multiply the accumulator contents by the real number 15.0
OUTD
V1400
Copy the result in the accumulator to V1400 and V1401.
4 0 E 0 0 0 0 0
7 (decimal)
x 1 5
1 0 5
4 0 E 0 0 0 0 0 (Accumulator)
X 4 1 7 0 0 0 0 0 (MULR)
Acc. 4 2 D 2 0 0 0 0
V1401
V1400
4 2 D 2 0 0 0 0 (Hex number)
Real Value
8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1
Acc. 0 1 0 0 0 0 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Sign Bit
Exponent (8 bits)
Mantissa (23 bits)
128 + 4 + 1 = 133
133 - 127 = 6
Implies 2 (exp 6)
1.101001 x 2 (exp 6) = 1101001. binary= 105 decimal

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.

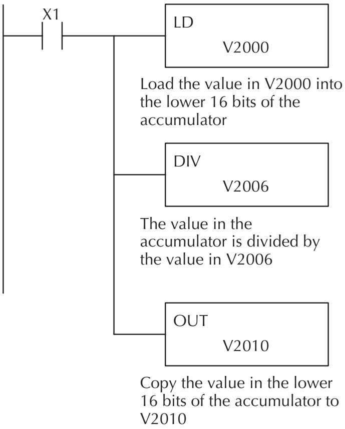


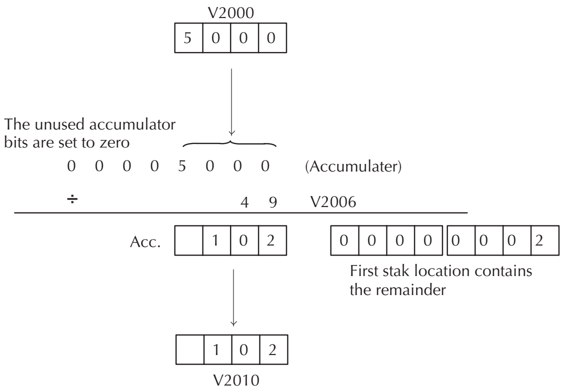
Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0 – 9999

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:** Status flags are valid only until another instruction uses the same flag.*

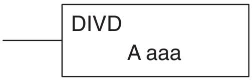
In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the accumulator will be divided by the value in V2006 using the Divide instruction. The value in the accumulator is copied to V2010 using the Out instruction.


X1
LD
V2000
Load the value in V2000 into the lower 16 bits of the accumulator
DIV
V2006
The value in the accumulator is divided by the value in V2006
OUT
V2010
Copy the value in the lower 16 bits of the accumulator to V2010


V2000
5 0 0 0
The unused accumulator bits are set to zero
0 0 0 0 5 0 0 0
(Accumulater)
÷
4 9
V2006
Acc.
1 0 2
0 0 0 0 0 0 0 2
First stak location contains the remainder
1 0 2
V2010

Divide Double (DIVD)

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.

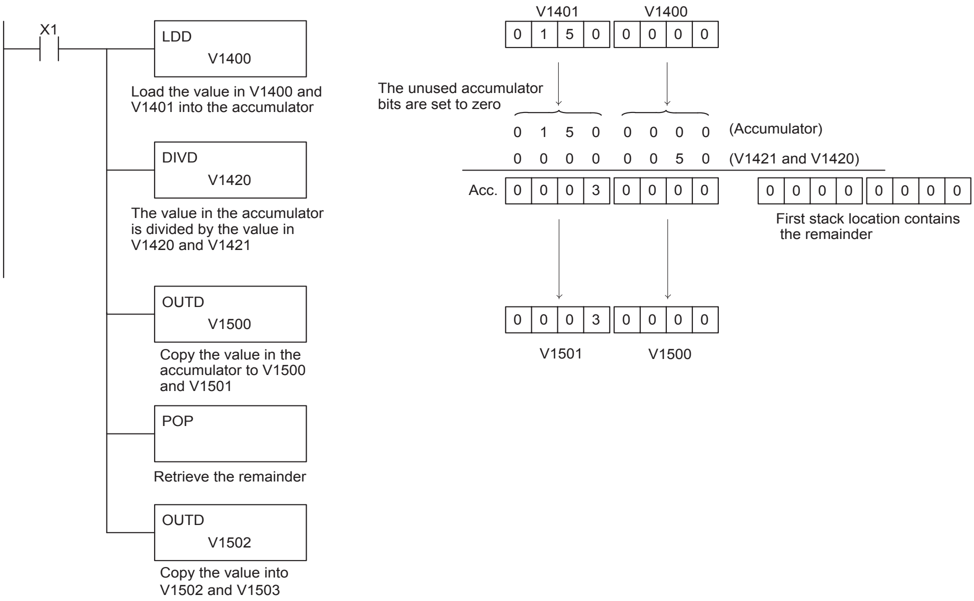


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map

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 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.


X1
LDD
V1400
Load the value in V1400 and V1401 into the accumulator
DIVD
V1420
The value in the accumulator is divided by the value in V1420 and V1421
OUTD
V1500
Copy the value in the accumulator to V1500 and V1501
POP
Retrieve the remainder
OUTD
V1502
Copy the value into V1502 and V1503
V1401
V1400
0 1 5 0 0 0 0 0
The unused accumulator bits are set to zero
0 1 5 0 0 0 0 0
(Accumulator)
0 0 0 0 0 0 5 0
(V1421 and V1420)
Acc.
0 0 0 3 0 0 0 0
0 0 0 0 0 0 0 0
First stack location contains the remainder
0 0 0 3 0 0 0 0
V1501
V1500

Divide Real (DIVR)

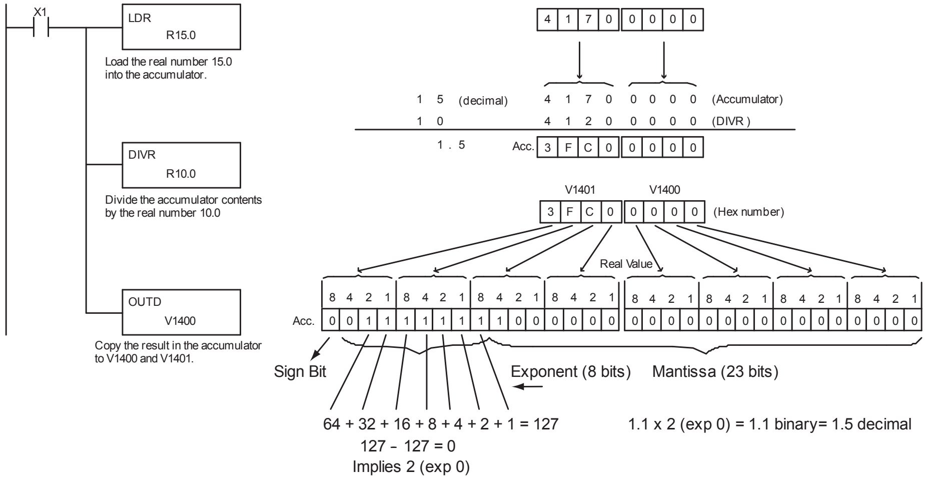
The Divide Real instruction divides a real number in the accumulator by either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must be Real data type (IEEE floating point format).

DIVR
A aaa

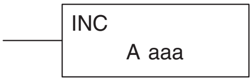
Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Real Constant	R	$-3.402823E^{+38}$ to + $+3.402823E^{+38}$

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.
SP74	On anytime a floating point math operation results in an underflow error.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

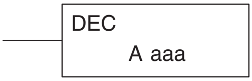

X1
LDR
R15.0
Load the real number 15.0 into the accumulator.
DIVR
R10.0
Divide the accumulator contents by the real number 10.0
OUTD
V1400
Copy the result in the accumulator to V1400 and V1401.
4 1 7 0 0 0 0 0
1 5 (decimal) 4 1 7 0 0 0 0 0 (Accumulator)
1 0 4 1 2 0 0 0 0 0 (DIVR)
1 . 5 Acc. 3 F C 0 0 0 0 0
V1401 V1400
3 F C 0 0 0 0 0 (Hex number)
Real Value
8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1
Acc. 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Sign Bit
Exponent (8 bits)
Mantissa (23 bits)
64 + 32 + 16 + 8 + 4 + 2 + 1 = 127
127 – 127 = 0
Implies 2 (exp 0)
1.1 x 2 (exp 0) = 1.1 binary= 1.5 decimal

Increment (INC)



The Increment instruction increments a BCD value in a specified V-memory location by “1” each time the instruction is executed.

Decrement (DEC)



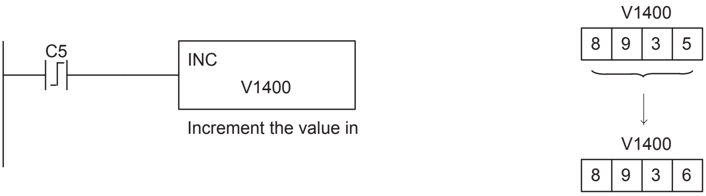
The Decrement instruction decrements a BCD value in a specified V-memory location by “1” each time the instruction is executed.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map

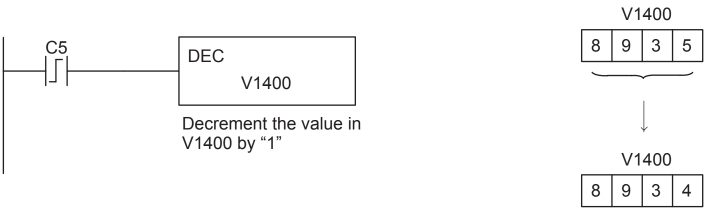
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP75	On when a BCD instruction is executed and a NON – BCD number was encountered.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

In the following increment example, when C5 makes an Off-to-On transition the value in V1400 increases by one.

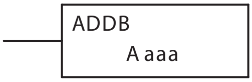


In the following decrement example, when C5 makes an Off-to-On transition the value in V1400 is decreased by one.



Add Binary (ADDB)

Add Binary is a 16-bit instruction that adds the binary value in the lower 16 bits of the accumulator with a binary value (Aaaa), which is either a V-memory location or a 16-bit constant. The result can be up to 32 bits and resides in the accumulator.

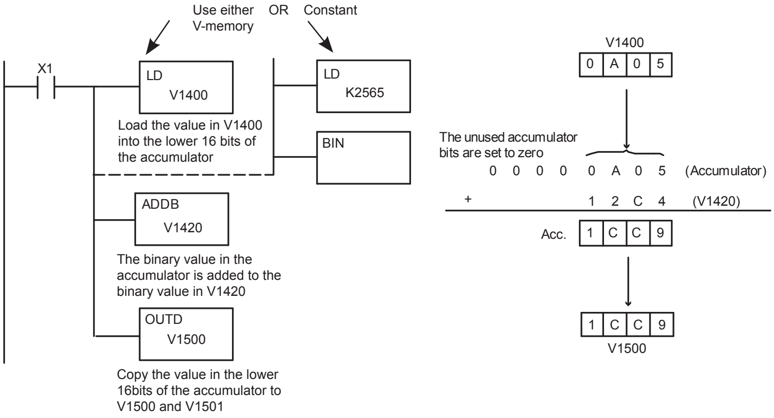


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0-FFFF

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 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 will be loaded into the accumulator using the Load instruction. The binary value in the accumulator will be added to the binary value in V1420 using the Add Binary instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.


Use either V-memory
OR
Constant
X1
LD
V1400
LD
K2565
BIN
Load the value in V1400 into the lower 16 bits of the accumulator
ADDB
V1420
The binary value in the accumulator is added to the binary value in V1420
OUTD
V1500
Copy the value in the lower 16bits of the accumulator to V1500 and V1501
V1400
0 A 0 5
The unused accumulator bits are set to zero
0 0 0 0 0 A 0 5 (Accumulator)
+ 1 2 C 4 (V1420)
Acc. 1 C C 9
1 C C 9
V1500

Add Binary Double (ADDBD)

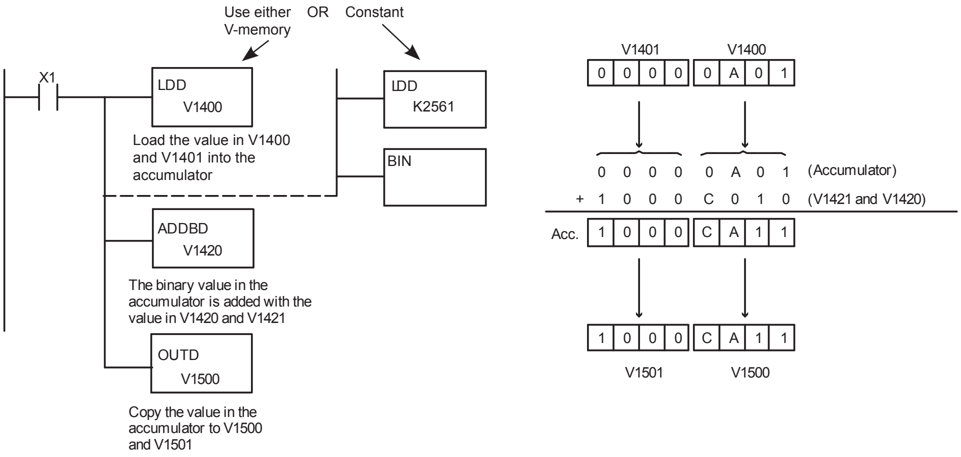
Add Binary Double is a 32-bit instruction that adds the binary value in the accumulator with the value (Aaaa), which is either two consecutive V-memory locations or an 8-digit (max.) binary constant. The result resides in the accumulator.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
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.
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.

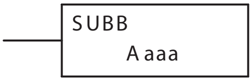
***NOTE:** Status flags are valid only until another instruction uses the same flag.*

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.


Use either V-memory
OR
Constant
X1
LDD
V1400
Load the value in V1400 and V1401 into the accumulator
LDD
K2561
BIN
ADDBD
V1420
The binary value in the accumulator is added with the value in V1420 and V1421
OUTD
V1500
Copy the value in the accumulator to V1500 and V1501
V1401
V1400
0 0 0 0 0 A 0 1
0 0 0 0 0 A 0 1 (Accumulator)
+ 1 0 0 0 C 0 1 0 (V1421 and V1420)
Acc. 1 0 0 0 C A 1 1
1 0 0 0 C A 1 1
V1501
V1500

Subtract Binary (SUBB)

Subtract Binary is a 16-bit instruction that subtracts the binary value (Aaaa), which is either a V-memory location or a 4-digit (max.) binary constant, from the binary value in the accumulator. The result resides in the accumulator.

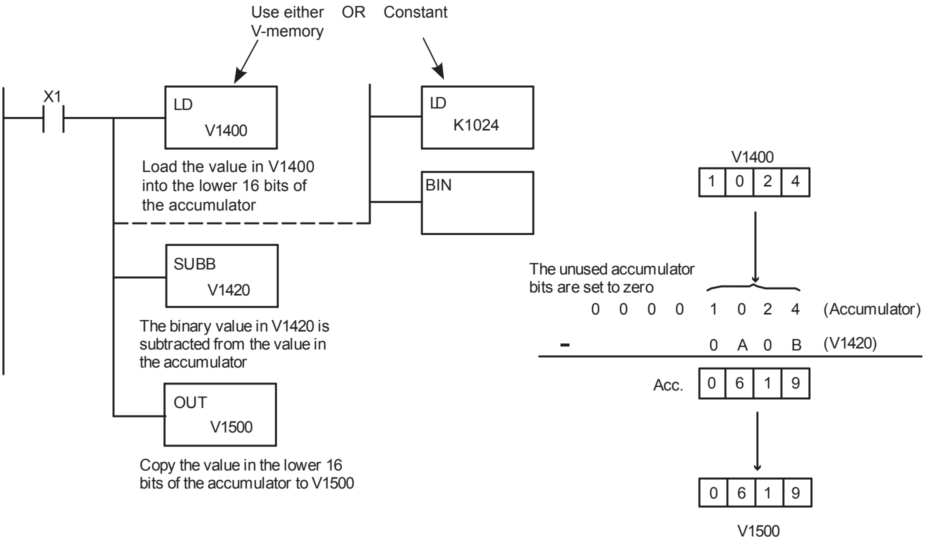


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0-FFFF

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.
SP70	On anytime the value in the accumulator is negative.

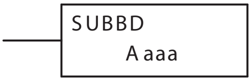
***NOTE:** Status flags are valid only until another instruction uses the same flag.*

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 using the Out instruction.


Use either V-memory
OR
Constant
X1
LD
V1400
LD
K1024
BIN
Load the value in V1400 into the lower 16 bits of the accumulator
SUBB
V1420
The binary value in V1420 is subtracted from the value in the accumulator
OUT
V1500
Copy the value in the lower 16 bits of the accumulator to V1500
V1400
1 0 2 4
The unused accumulator bits are set to zero
0 0 0 0 1 0 2 4 (Accumulator)
– 0 A 0 B (V1420)
Acc. 0 6 1 9
0 6 1 9
V1500

Subtract Binary Double (SUBBD)

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.

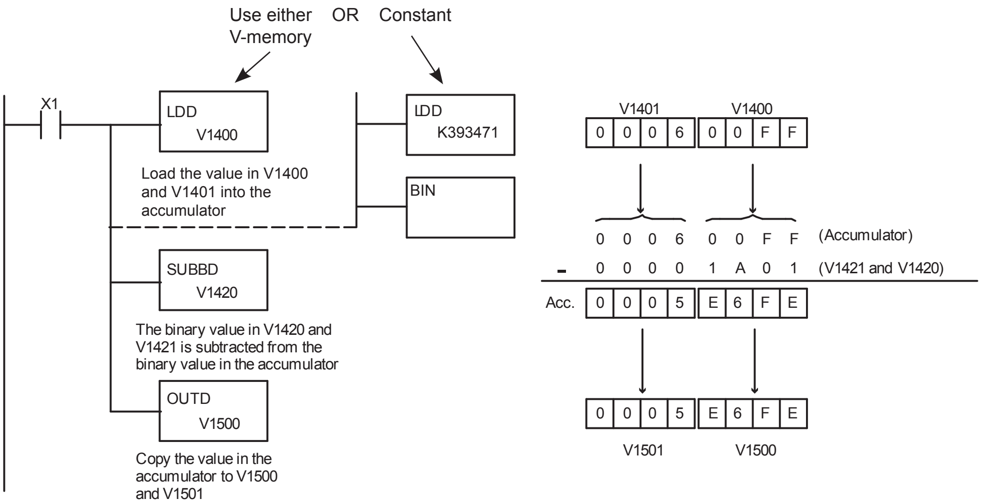


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0-FFFFFFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP65	On when the 32-bit subtraction instruction results in a borrow.
SP70	On anytime the value in the accumulator is negative.

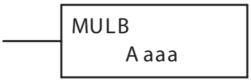
NOTE: *Status flags are valid only until another instruction uses the same flag.*

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.


Use either V-memory
OR
Constant
X1
LDD
V1400
LDD
K393471
BIN
Load the value in V1400 and V1401 into the accumulator
SUBBD
V1420
The binary value in V1420 and V1421 is subtracted from the binary value in the accumulator
OUTD
V1500
Copy the value in the accumulator to V1500 and V1501
V1401
V1400
0 0 0 6 0 0 F F
0 0 0 6 0 0 F F (Accumulator)
- 0 0 0 0 1 A 0 1 (V1421 and V1420)
Acc. 0 0 0 5 E 6 F E
0 0 0 5 E 6 F E
V1501
V1500

Multiply Binary (MULB)

Multiply Binary is a 16-bit instruction that multiplies the binary value (Aaaa), which is either a V-memory location or a 4-digit (max.) binary constant, by the binary value in the accumulator. The result can be up to 32 bits and resides in the accumulator.

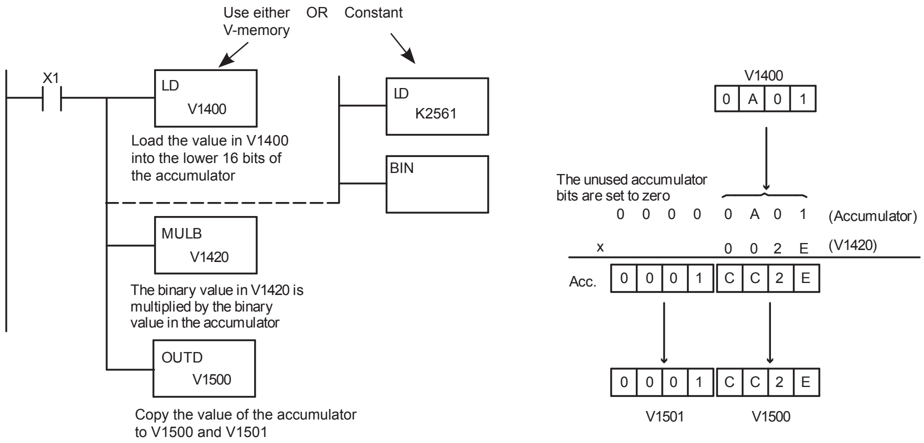


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0-FFFF

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.

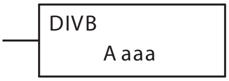
NOTE: *Status flags are valid only until another instruction uses the same flag.*

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 using the Out instruction.


Use either V-memory
OR
Constant
X1
LD
V1400
LD
K2561
BIN
Load the value in V1400 into the lower 16 bits of the accumulator
MULB
V1420
The binary value in V1420 is multiplied by the binary value in the accumulator
OUTD
V1500
Copy the value of the accumulator to V1500 and V1501
V1400
0 A 0 1
The unused accumulator bits are set to zero
0 0 0 0 0 A 0 1
(Accumulator)
x
0 0 2 E
(V1420)
Acc.
0 0 0 1 C C 2 E
0 0 0 1 C C 2 E
V1501
V1500

Divide Binary (DIVB)

Divide Binary is a 16-bit instruction that divides the binary value in the accumulator by a binary value (Aaaa), which is either a V-memory location or a 16-bit (max.) binary constant. The first part of the quotient resides in the accumulator and the remainder resides in the first stack location.

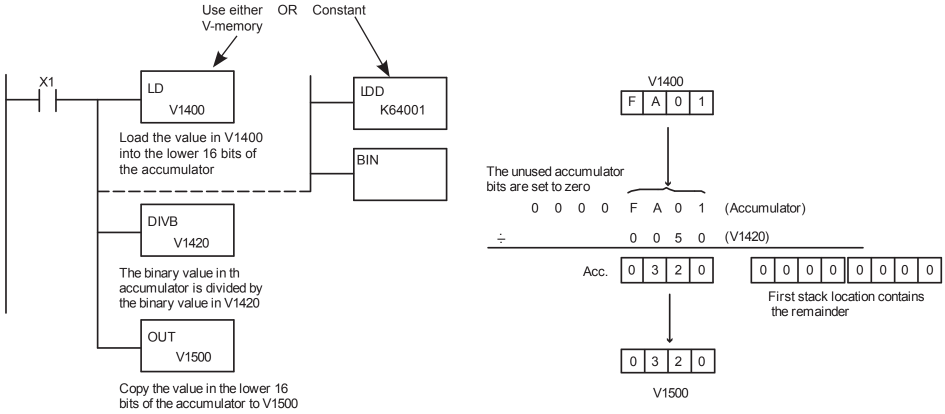


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0-FFFF

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.

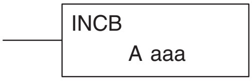
NOTE: *Status flags are valid only until another instruction uses the same flag.*

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 instruction.


Use either
V-memory
OR
Constant
X1
LD
V1400
LDD
K64001
BIN
Load the value in V1400 into the lower 16 bits of the accumulator
DIVB
V1420
The binary value in th accumulator is divided by the binary value in V1420
OUT
V1500
Copy the value in the lower 16 bits of the accumulator to V1500
V1400
F A 0 1
The unused accumulator bits are set to zero
0 0 0 0 F A 0 1 (Accumulator)
÷ 0 0 5 0 (V1420)
Acc. 0 3 2 0
0 0 0 0 0 0 0 0
First stack location contains the remainder
0 3 2 0
V1500

Increment Binary (INCB)

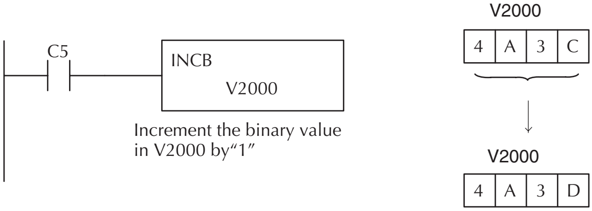
The Increment Binary instruction increments a binary value in a specified V-memory location by "1" each time the instruction is executed.



Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.

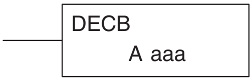
In the following example when C5 is on, the binary value in V2000 is increased by 1.



Increment the binary value in V2000 by"1"

Decrement Binary (DECB)

The Decrement Binary instruction decrements a binary value in a specified V-memory location by "1" each time the instruction is executed.

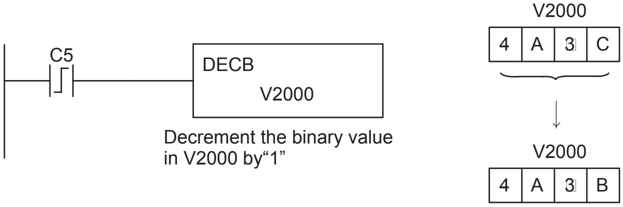


Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.

NOTE: *Status flags are valid only until another instruction uses the same flag.*

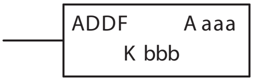
In the following example when C5 is on, the value in V2000 is decreased by 1.



Decrement the binary value in V2000 by"1"

Add Formatted (ADDF)

Add Formatted is a 32-bit instruction that adds the BCD value in the accumulator with the BCD value (Aaaa) which is a range of discrete bits. The specified range (Kbbb) can be 1 to 32 consecutive bits. The result resides in the accumulator.

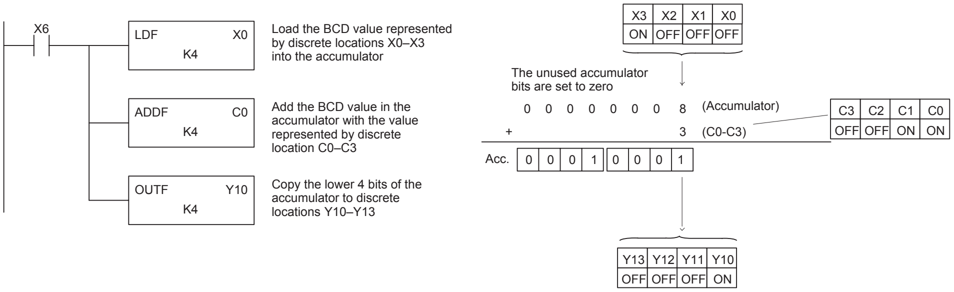


Operand Data Type		D4-454 Range	
	A	aaa	bbb
Inputs	X	0 – 1777	- -
Outputs	Y	0 – 1777	- -
Control Relays	C	0 – 3777	- -
Stage Bits	S	0 – 1777	- -
Timer Bits	T	0 – 377	- -
Counter Bits	CT	0 – 377	- -
Special Relays	SP	0 – 777	- -
Global I/O	GX	0 – 3777	- -
Constant	K	- -	1 – 32

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
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.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

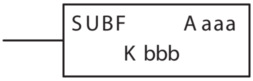
***NOTE:** Status flags are valid only until another instruction uses the same flag.*

In the following example, when X6 is on, the BCD value formed by discrete locations X0 – X3 is loaded into the accumulator using the LDF instruction. The BCD value formed by discrete locations C0 – C3 is added to the value in the accumulator using the ADDF instruction. The value in the lower four bits of the accumulator is copied to Y10 – Y13 using the OUTF instruction.


X6
LDF X0
K4
Load the BCD value represented by discrete locations X0–X3 into the accumulator
ADDF C0
K4
Add the BCD value in the accumulator with the value represented by discrete location C0–C3
OUTF Y10
K4
Copy the lower 4 bits of the accumulator to discrete locations Y10–Y13
X3 X2 X1 X0
ON OFF OFF OFF
The unused accumulator bits are set to zero
0 0 0 0 0 0 0 8 (Accumulator)
+ 3 (C0-C3)
C3 C2 C1 C0
OFF OFF ON ON
Acc. 0 0 0 1 0 0 0 1
Y13 Y12 Y11 Y10
OFF OFF OFF ON

Subtract Formatted (SUBF)

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.

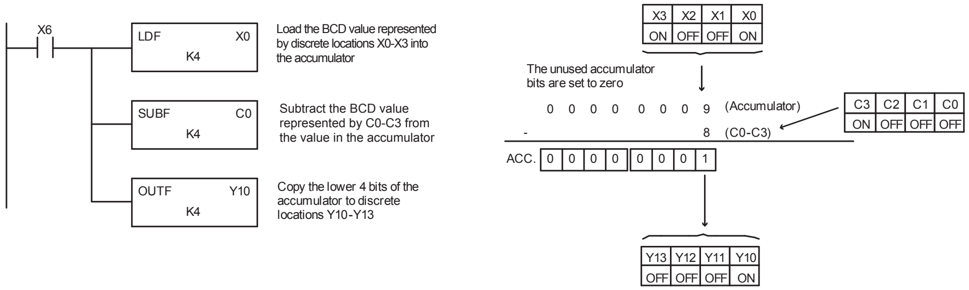


Operand Data Type		D4-454 Range	
	A	aaa	bbb
Inputs	X	0 – 1777	– –
Outputs	Y	0 – 1777	– –
Control Relays	C	0 – 3777	– –
Stage Bits	S	0 – 1777	– –
Timer Bits	T	0 – 377	– –
Counter Bits	CT	0 – 377	– –
Special Relays	SP	0 – 777	– –
Global I/O	GX	0 – 3777	– –
Constant	K	– –	1 – 32

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP65	On when the 32 bit subtraction instruction results in a borrow
SP70	On any time the value in the accumulator is negative.
SP73	On when a signed addition or subtraction results in an incorrect sign bit.
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 BCD value formed by discrete locations X0 – X3 is loaded into the accumulator using the LDF instruction. The BCD value formed by discrete location C0 – C3 is subtracted from the BCD value in the accumulator using the SUBF instruction. The value in the lower four bits of the accumulator is copied to Y10 – Y13 using the OUTF instruction.



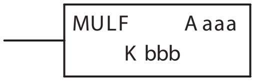
Load the BCD value represented by discrete locations X0-X3 into the accumulator

Subtract the BCD value represented by C0-C3 from the value in the accumulator

Copy the lower 4 bits of the accumulator to discrete locations Y10-Y13

The unused accumulator bits are set to zero

Multiply Formatted (MULF)



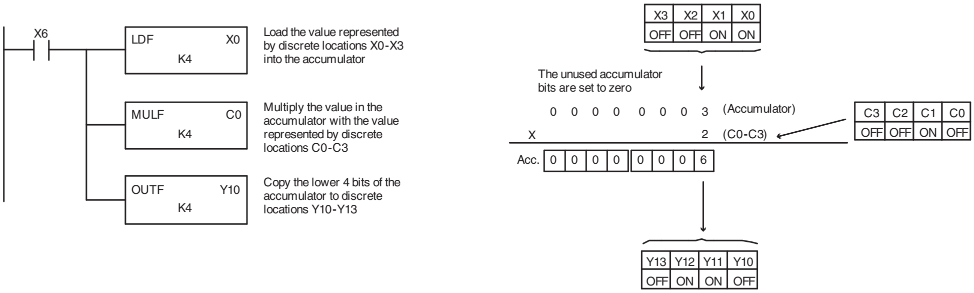
Multiply Formatted is a 16-bit instruction that multiplies the BCD value in the accumulator by the BCD value (Aaaa) which is a range of discrete bits. The specified range (Kbbb) can be 1 to 16 consecutive bits. The result resides in the accumulator.

Operand Data Type		D4-454 Range	
	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	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 any time the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

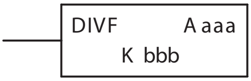
NOTE: *Status flags are valid only until another instruction uses the same flag.*

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.


X6
LDF X0
K4
Load the value represented by discrete locations X0-X3 into the accumulator
MULF C0
K4
Multiply the value in the accumulator with the value represented by discrete locations C0-C3
OUTF Y10
K4
Copy the lower 4 bits of the accumulator to discrete locations Y10-Y13
X3 X2 X1 X0
OFF OFF ON ON
The unused accumulator bits are set to zero
0 0 0 0 0 0 0 3 (Accumulator)
X 2 (C0-C3)
C3 C2 C1 C0
OFF OFF ON OFF
Acc. 0 0 0 0 0 0 0 6
Y13 Y12 Y11 Y10
OFF ON ON OFF

Divide Formatted (DIVF)

Divide Formatted is a 16-bit instruction that divides the BCD value in the accumulator 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

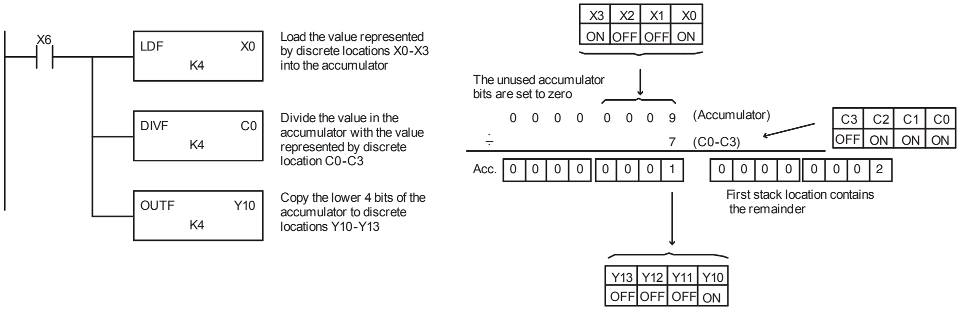


Operand Data Type		D4-454 Range	
	A	aaa	bbb
Inputs	X	0 – 1777	- -
Outputs	Y	0 – 1777	- -
Control Relays	C	0 – 3777	- -
Stage Bits	S	0 – 1777	- -
Timer Bits	T	0 – 377	- -
Counter Bits	CT	0 – 377	- -
Special Relays	P	0 – 777	- -
Global I/O	X	0 – 3777	- -
Constant	K	- -	1 – 16

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 any time 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.



Load the value represented by discrete locations X0-X3 into the accumulator

Divide the value in the accumulator with the value represented by discrete location C0-C3

Copy the lower 4 bits of the accumulator to discrete locations Y10-Y13

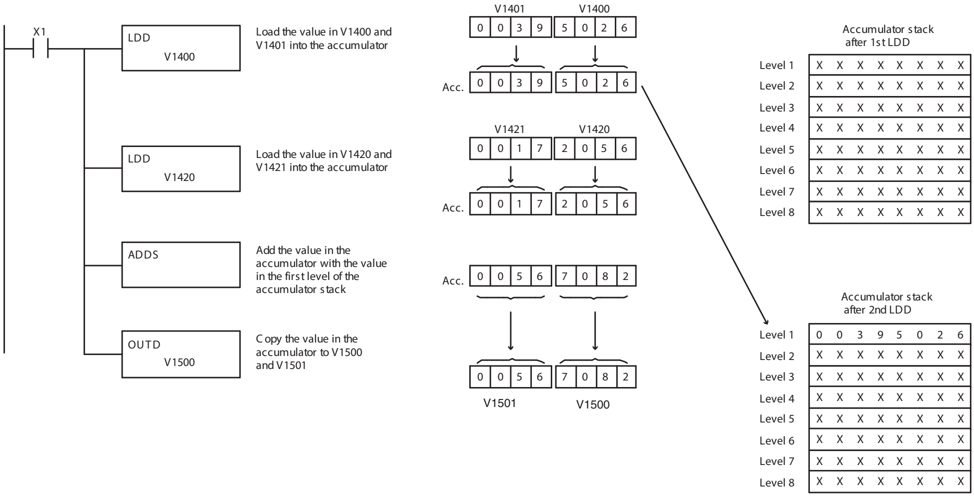
Add Top of Stack (ADDS)

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.
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.
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 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.



Load the value in V1400 and V1401 into the accumulator

Load the value in V1420 and V1421 into the accumulator

Add the value in the accumulator with the value in the first level of the accumulator stack

Copy the value in the accumulator to V1500 and V1501

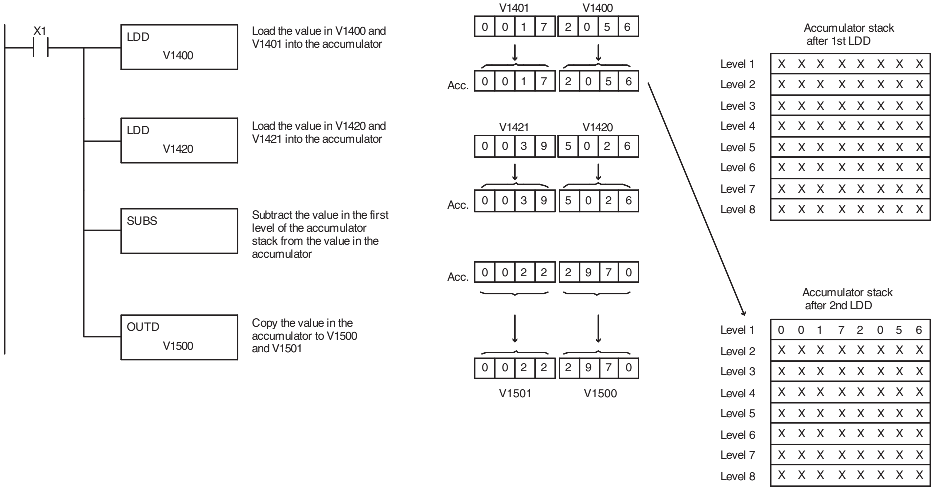
Subtract Top of Stack (SUBS)

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.
SP65	On when the 32 bit subtraction instruction results in a borrow.
SP70	On anytime the value in the accumulator is negative.
SP73	On when a signed addition or subtraction results in an incorrect sign bit.
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 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.



Load the value in V1400 and V1401 into the accumulator

Load the value in V1420 and V1421 into the accumulator

Subtract the value in the first level of the accumulator stack from the value in the accumulator

Copy the value in the accumulator to V1500 and V1501

Accumulator stack after 1st LDD

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

Accumulator stack after 2nd LDD

Level 1	0 0 1 7 2 0 5 6
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

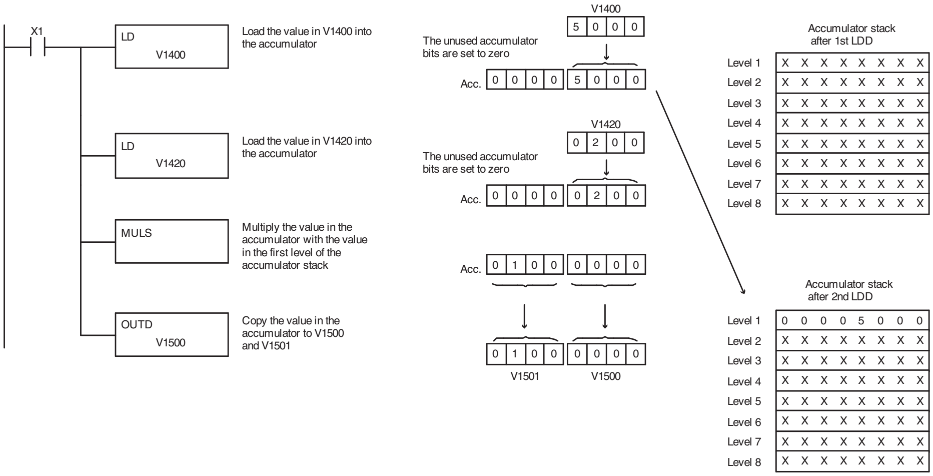
Multiply Top of Stack (MULS)

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 any time 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 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.



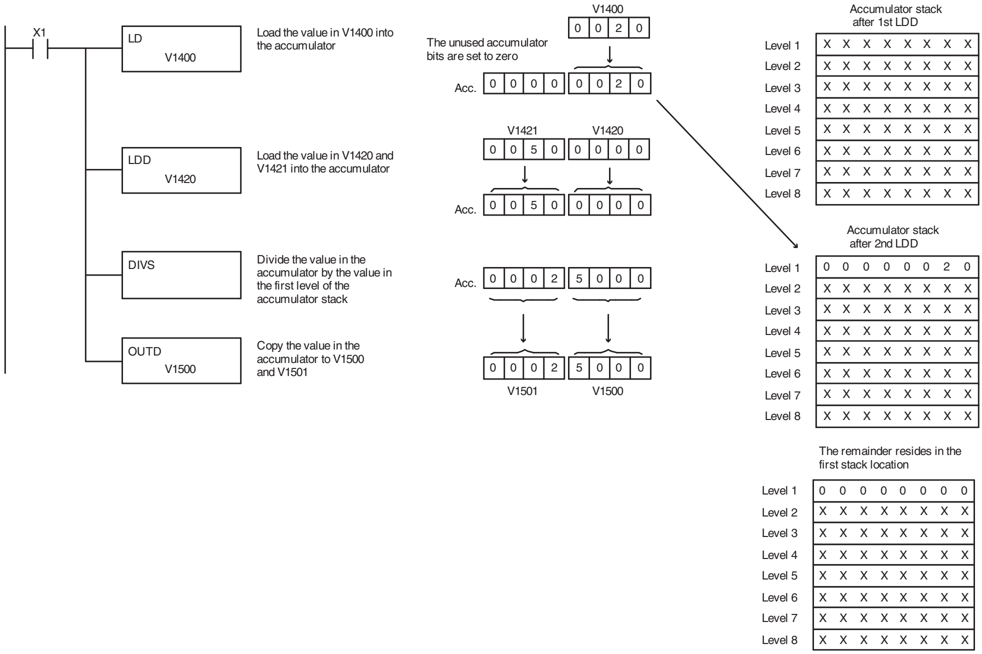
Divide by Top of Stack (DIVS)

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 any time 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 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.



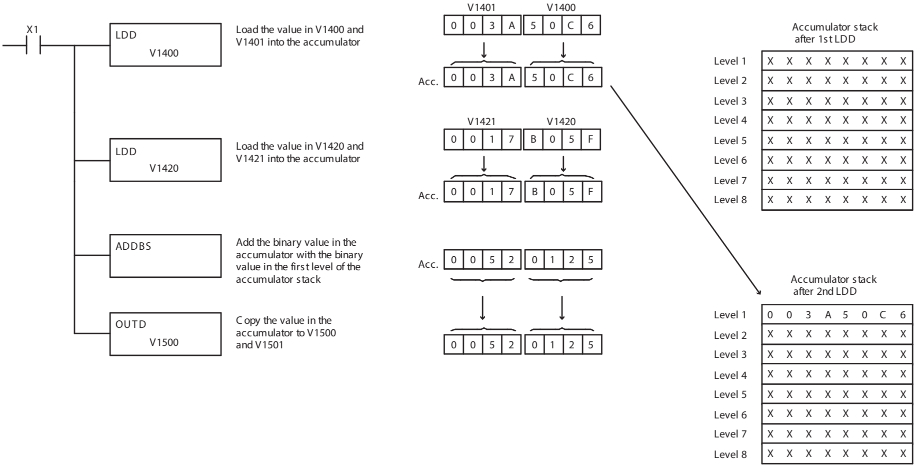
Add Binary Top of Stack (ADDBS)

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

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
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.

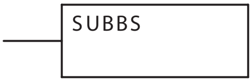
NOTE: *Status flags are valid only until another instruction uses the same flag.*

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 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.



Subtract Binary Top of Stack (SUBBS)

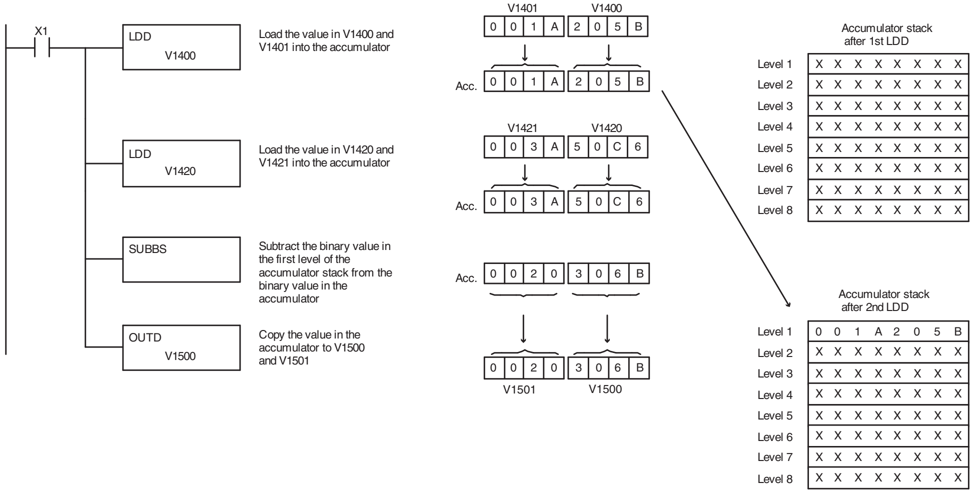
Subtract Binary Top of Stack is a 32-bit instruction that subtracts the binary value in the first level of the accumulator stack from the binary value in the accumulator. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack locations are moved up one level.



Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP65	On when the 32-bit subtraction instruction results in a borrow.
SP70	On any time 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 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.


X1
LDD
V1400
Load the value in V1400 and V1401 into the accumulator
LDD
V1420
Load the value in V1420 and V1421 into the accumulator
SUBBS
Subtract the binary value in the first level of the accumulator stack from the binary value in the accumulator
OUTD
V1500
Copy the value in the accumulator to V1500 and V1501
V1401
V1400
0 0 1 A 2 0 5 B
Acc.
0 0 1 A 2 0 5 B
V1421
V1420
0 0 3 A 5 0 C 6
Acc.
0 0 3 A 5 0 C 6
Acc.
0 0 2 0 3 0 6 B
0 0 2 0 3 0 6 B
V1501
V1500
Accumulator stack after 1st LDD
Accumulator stack after 2nd LDD

Accumulator stack after 1st LDD

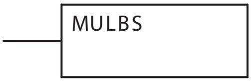
Level 1	X	X	X	X	X	X	X	X
Level 2	X	X	X	X	X	X	X	X
Level 3	X	X	X	X	X	X	X	X
Level 4	X	X	X	X	X	X	X	X
Level 5	X	X	X	X	X	X	X	X
Level 6	X	X	X	X	X	X	X	X
Level 7	X	X	X	X	X	X	X	X
Level 8	X	X	X	X	X	X	X	X

Accumulator stack after 2nd LDD

Level 1	0	0	1	A	2	0	5	B
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

Multiply Binary Top of Stack (MULBS)

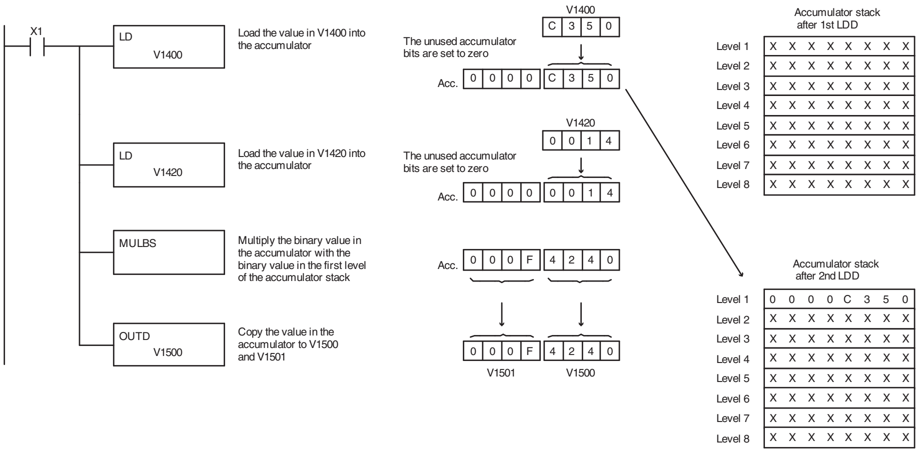
Multiply Binary Top of Stack is a 16-bit instruction that multiplies the 16-bit binary value in the first level of the accumulator stack by the 16-bit binary value in the accumulator. The result resides in the accumulator and can be 32 bits (8 digits max.). The value in the first level of the accumulator 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 any time 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 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.



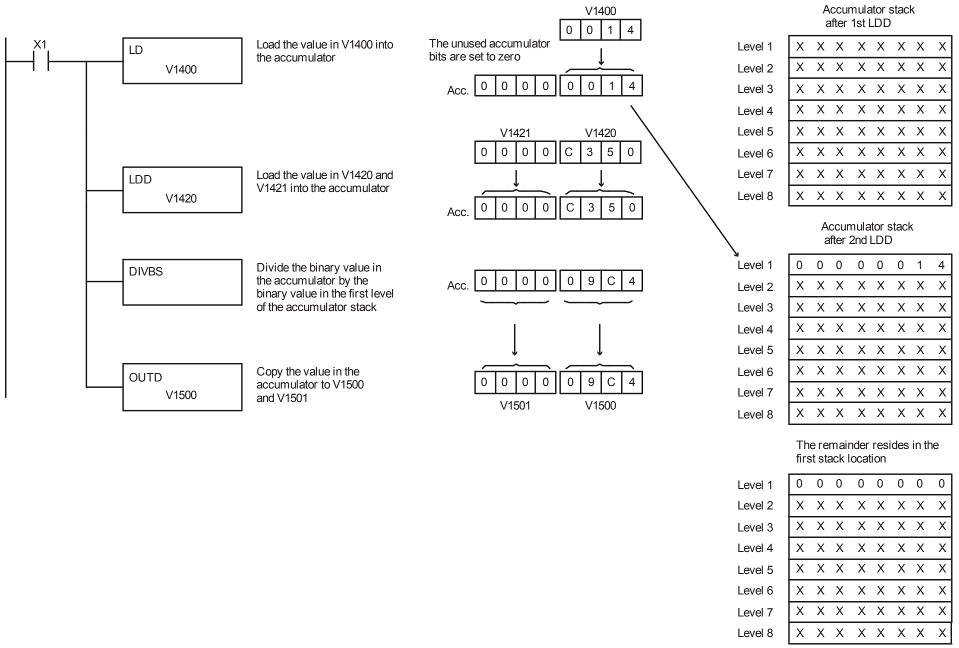
Divide Binary by Top OF Stack (DIVBS)

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 any time 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 value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction also, 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.



Load the value in V1400 into the accumulator

The unused accumulator bits are set to zero

Load the value in V1420 and V1421 into the accumulator

Divide the binary value in the accumulator by the binary value in the first level of the accumulator stack

Copy the value in the accumulator to V1500 and V1501

Accumulator stack after 1st LDD

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

Accumulator stack after 2nd LDD

Level 1	0	0	0	0	0	0	1	4
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

The remainder resides in the first stack location

Level 1	0	0	0	0	0	0	0	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

Transcendental Functions

The D4-454 CPU features special numerical functions to complement its 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 an invalid floating point number
SP73	On when a signed addition or subtraction results in an incorrect sign bit.
SP75	On when a real number instruction is executed and a non-real number encountered.

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 must be Real data type (IEEE floating point 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 must be Real data type (IEEE floating point 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 must be Real data type (IEEE floating point 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 must be Real data type (IEEE floating point 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 must be Real data type (IEEE floating point format).

Arc Tangent Real (ATANR)

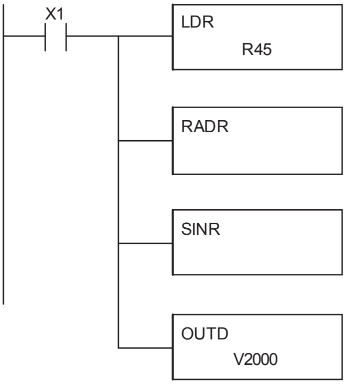
The Arc Tangent Real instruction takes the inverse tangent of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result must be Real data type (IEEE floating point format).

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 must be Real data type (IEEE floating point 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.*

The following example takes the sine of 45 degrees. Since these transcendental functions operate only on real numbers, we do an 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.



	Accumulator contents (viewed as real number)
Load the real number 45 into the accumulator.	45.000000
Convert the degrees into radians, leaving the result in the accumulator.	0.7358981
Take the sine of the number in the accumulator, which is in radians.	0.7071067
Copy the value in the accumulator to V2000 and V2001.	0.7071067

Bit Operation Instructions

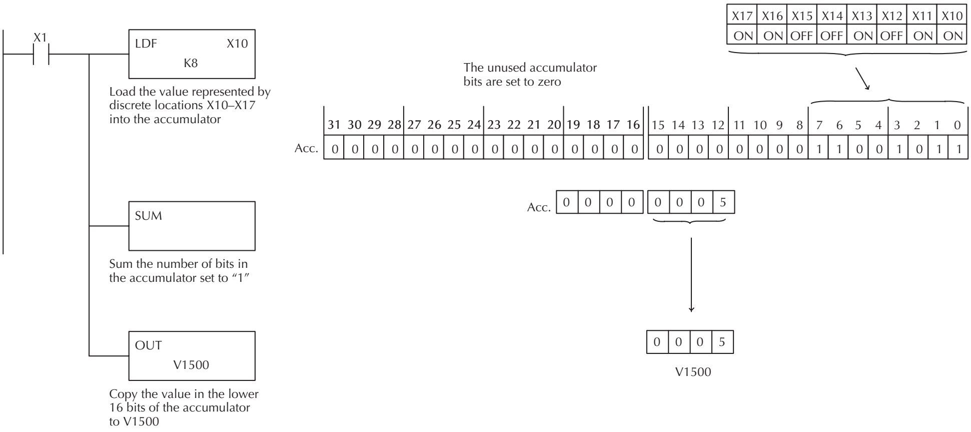
Sum (SUM)

The Sum instruction counts number of bits that are set to “1” in the accumulator. The HEX result resides in the accumulator.

Discrete Bit Flags	Description
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.

NOTE: *Status flags are valid only until another instruction uses the same flag.*


X1
LDF X10
K8
Load the value represented by discrete locations X10–X17 into the accumulator
SUM
Sum the number of bits in the accumulator set to “1”
OUT V1500
Copy the value in the lower 16 bits of the accumulator to V1500
X17 X16 X15 X14 X13 X12 X11 X10
ON ON OFF OFF ON OFF ON ON
The unused accumulator bits are set to zero
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc.
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 1
Acc.
0 0 0 0 0 0 0 5
0 0 0 5
V1500

Shift Left (SHFL)

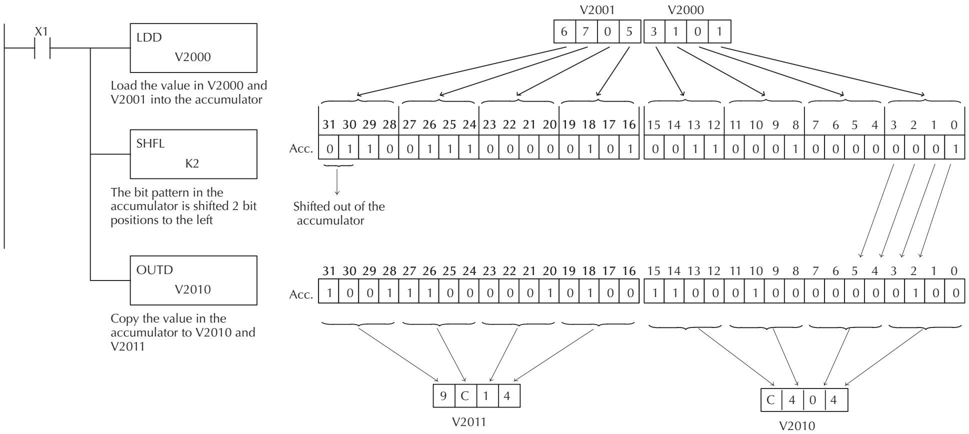
Shift Left is a 32-bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the left. The vacant positions are filled with zeros and the bits shifted out of the accumulator are discarded.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
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.
SP70	On anytime the value 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 bit pattern in the accumulator is shifted 2 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.

NOTE: *Status flags are valid only until another instruction uses the same flag.*



Load the value in V2000 and V2001 into the accumulator

The bit pattern in the accumulator is shifted 2 bit positions to the left

Copy the value in the accumulator to V2010 and V2011

Shift Right (SHFR)

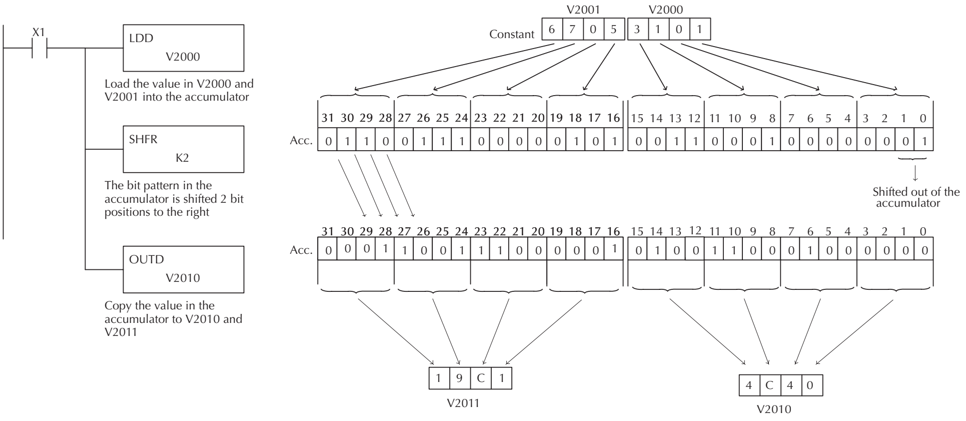
Shift Right is a 32-bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the right. The vacant positions are filled with zeros and the bits shifted out of the accumulator are lost.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Constant	K	1-32

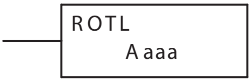
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.

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 bit pattern in the accumulator is shifted 2 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.

***NOTE:** Status flags are valid only until another instruction uses the same flag.*



Rotate Left (ROTL)



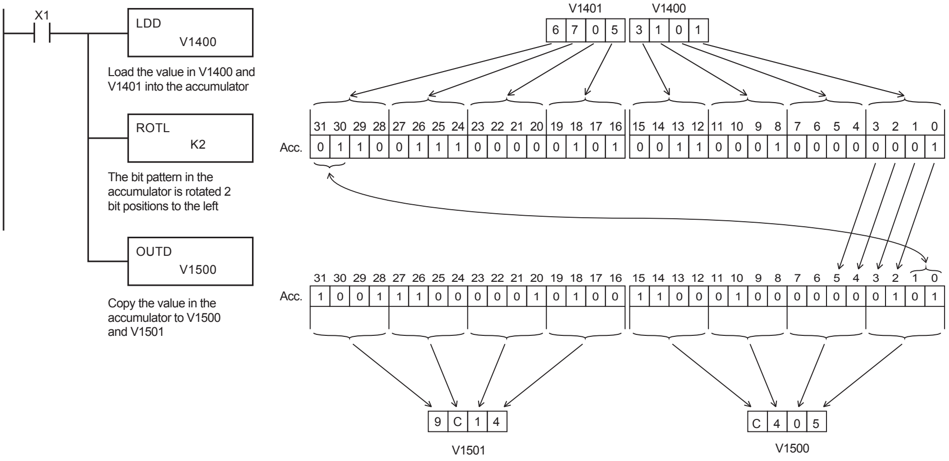
Rotate Left is a 32-bit instruction that rotates the bits in the accumulator a specified number (Aaaa) of places to the left.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
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.
SP70	On anytime the value in the accumulator is negative.

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.

NOTE: *Status flags are valid only until another instruction uses the same flag.*



Load the value in V1400 and V1401 into the accumulator

The bit pattern in the accumulator is rotated 2 bit positions to the left

Copy the value in the accumulator to V1500 and V1501

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	1	1	0	0	1	1	1	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	1	0	0	1	1	1	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1	0	1

Rotate Right (ROTR)

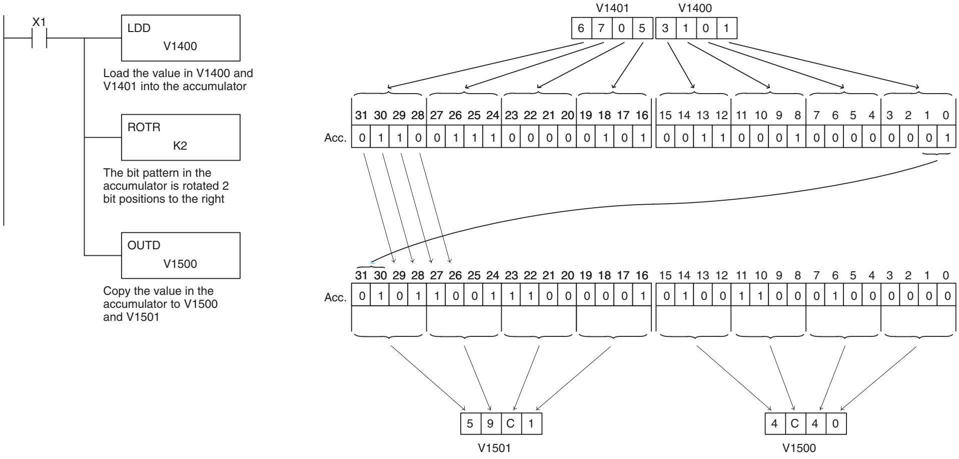
Rotate Right is a 32-bit instruction that rotates the bits in the accumulator a specified number (Aaaa) of places to the right.

```
ROTR
   A aaa
```

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
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.
SP70	On anytime the value in the accumulator is negative.

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.



Load the value in V1400 and V1401 into the accumulator

The bit pattern in the accumulator is rotated 2 bit positions to the right

Copy the value in the accumulator to V1500 and V1501

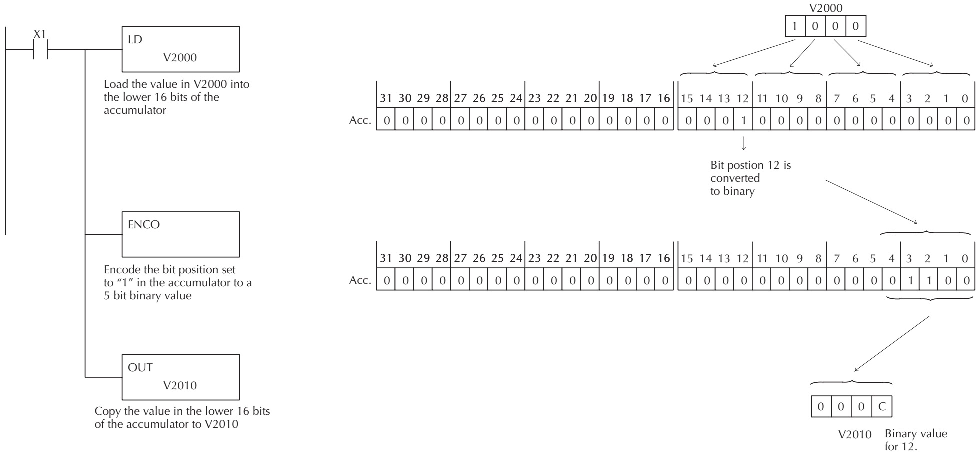
Encode (ENCO)

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.*

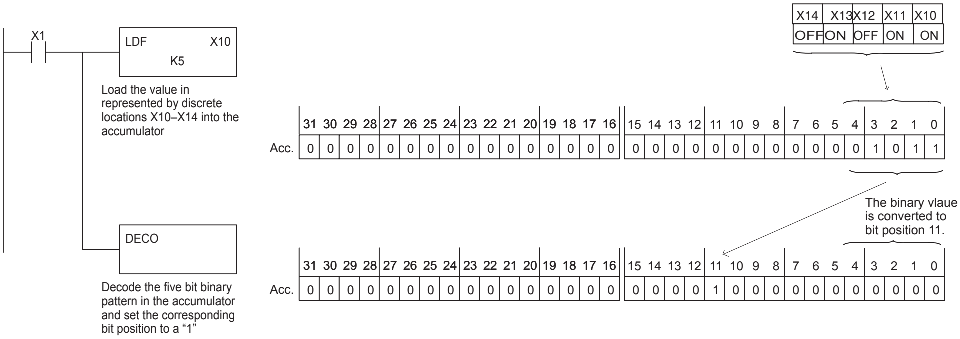
In the following example, when X1 is on, The value in V2000 is loaded into the accumulator using the Load instruction. The bit position set to a “1” in the accumulator is encoded to the corresponding 5 bit binary value using the Encode instruction. The value in the lower 16 bits of the accumulator is copied to V2010 using the Out instruction.


X1
LD
V2000
Load the value in V2000 into the lower 16 bits of the accumulator
ENCO
Encode the bit position set to “1” in the accumulator to a 5 bit binary value
OUT
V2010
Copy the value in the lower 16 bits of the accumulator to V2010
V2000
1 0 0 0
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
Bit postion 12 is converted to binary
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Acc. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
0 0 0 C
V2010
Binary value for 12.

Decode (DECO)

The Decode instruction decodes a 5-bit binary value of 0-31 (0 -1Fh) in the accumulator by setting the appropriate bit position to a 1. If the accumulator contains the value Fh (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.

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.


X1
LDF
X10
K5
Load the value in represented by discrete locations X10–X14 into the accumulator
DECO
Decode the five bit binary pattern in the accumulator and set the corresponding bit position to a “1”
Acc.
The binary vlaue is converted to bit position 11.

X14	X13	X12	X11	X10
OFF	ON	OFF	ON	ON

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Number Conversion Instructions (Accumulator)

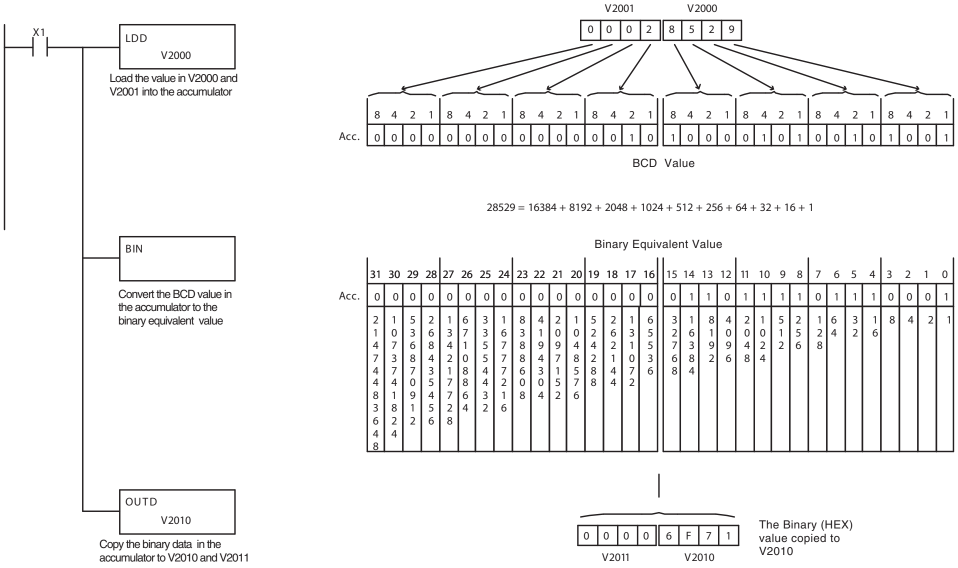
Binary (BIN)

The Binary instruction converts a BCD value in the accumulator to the equivalent binary, or decimal, value. The result resides in the accumulator.

BIN

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.

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.



Load the value in V2000 and V2001 into the accumulator

Convert the BCD value in the accumulator to the binary equivalent value

Copy the binary data in the accumulator to V2010 and V2011

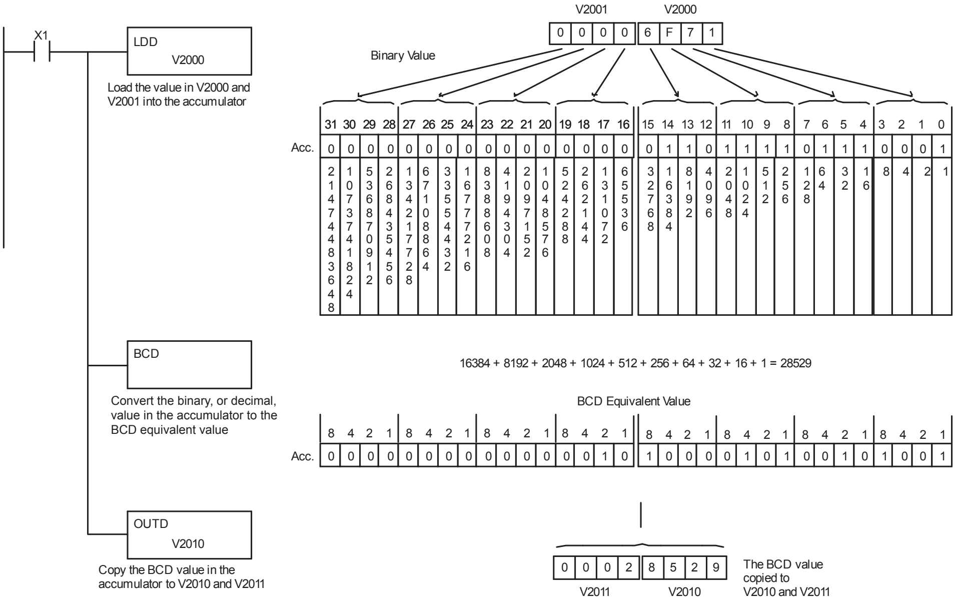
The Binary (HEX) value copied to V2010

Binary Coded Decimal (BCD)

The Binary Coded Decimal instruction converts a binary, or decimal, value in the accumulator to the equivalent BCD value. The result resides in the accumulator.

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 binary, or decimal, value in V2000 and V2001 is loaded into the accumulator using the Load Double instruction. The 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.



Load the value in V2000 and V2001 into the accumulator

16384 + 8192 + 2048 + 1024 + 512 + 256 + 64 + 32 + 16 + 1 = 28529

Convert the binary, or decimal, value in the accumulator to the BCD equivalent value

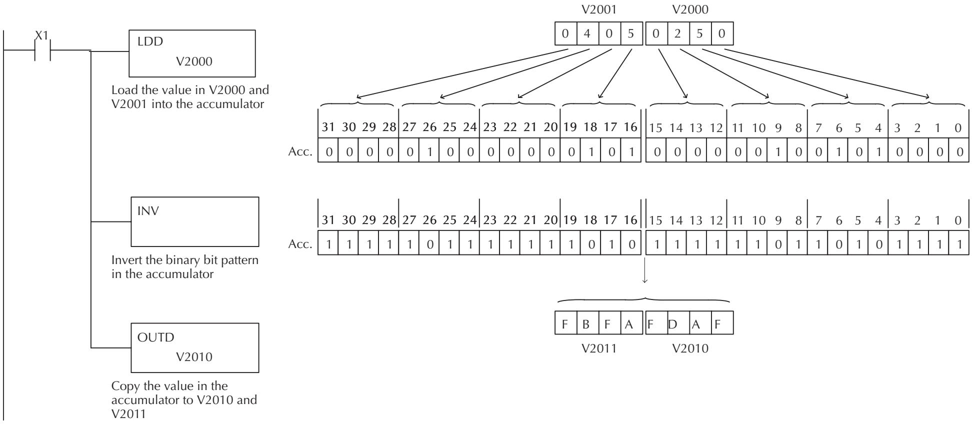
Copy the BCD value in the accumulator to V2010 and V2011

The BCD value copied to V2010 and V2011

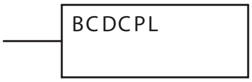
Invert (INV)

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.



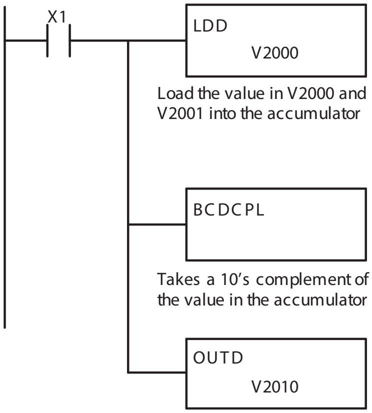
Ten's Complement (BCDCPL)



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 :

$$\begin{array}{c} 100000000 \\ -\ \text{accumulator} \\ \hline \text{10's complement value} \end{array}$$

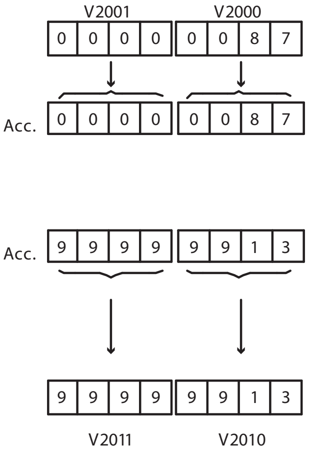
In the following example when X1 is on, the value in V2000 and V2001 is loaded into the accumulator. The 10's complement is taken for the 8 digit accumulator using the Ten's Complement instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



Load the value in V2000 and V2001 into the accumulator

Takes a 10's complement of the value in the accumulator

Copy the value in the accumulator to V2010 and V2011



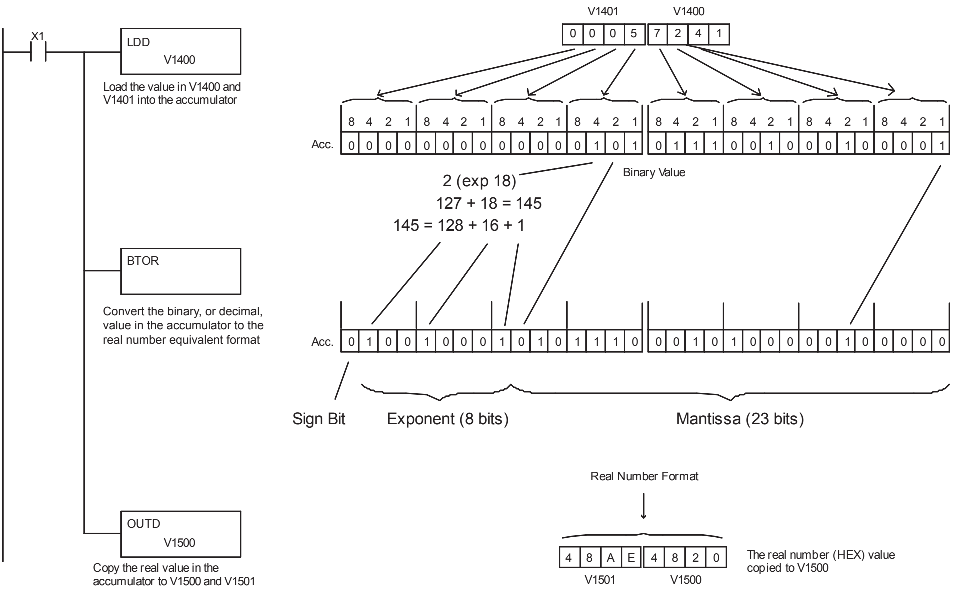
Binary to Real Conversion (BTOR)

The Binary-to-Real instruction converts a binary, or decimal, 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, or decimal, value in the accumulator to 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.



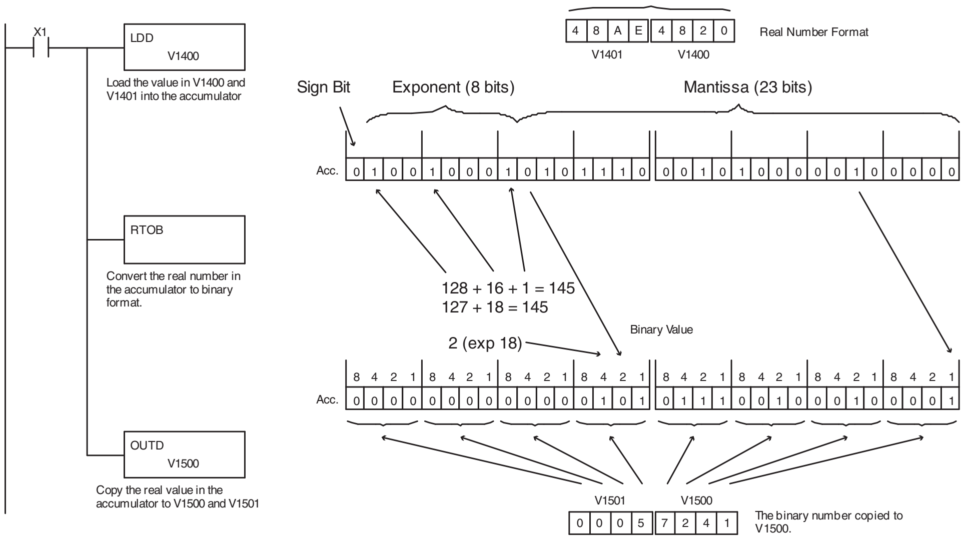
Real to Binary Conversion (RTOB)

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; -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 an invalid 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.

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.



Radian Real Conversion (RADR)

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.

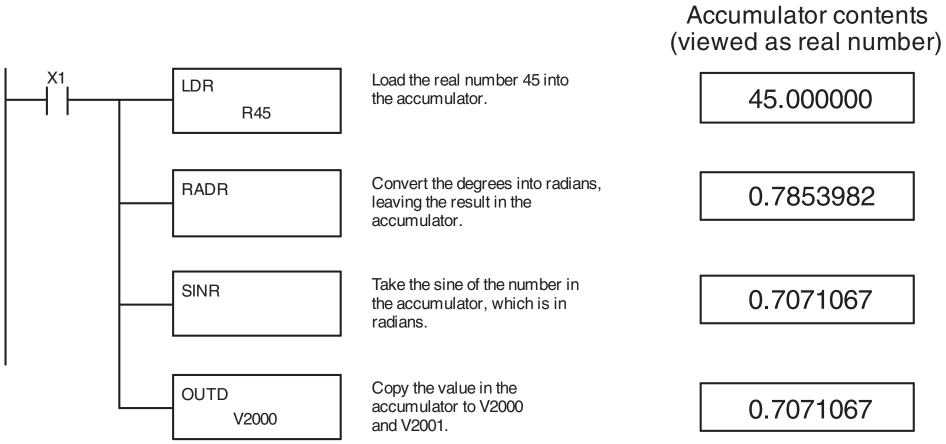
Degree Real Conversion (DEGR)

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.

The two instructions described above convert real numbers into the accumulator from degree format to radian format, and vice-versa. In degree format, a circle contains 360 degrees. In radian format, a circle contains 2π (about 6.28) radians. 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).

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 an invalid 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.

The following example takes the sine of 45 degrees. Since transcendental functions operate only on real numbers, we do an 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.


Accumulator contents
(viewed as real number)
X1
LDR
R45
Load the real number 45 into the accumulator.
45.000000
RADR
Convert the degrees into radians, leaving the result in the accumulator.
0.7853982
SINR
Take the sine of the number in the accumulator, which is in radians.
0.7071067
OUTD
V2000
Copy the value in the accumulator to V2000 and V2001.
0.7071067

ASCII to HEX (ATH)

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.

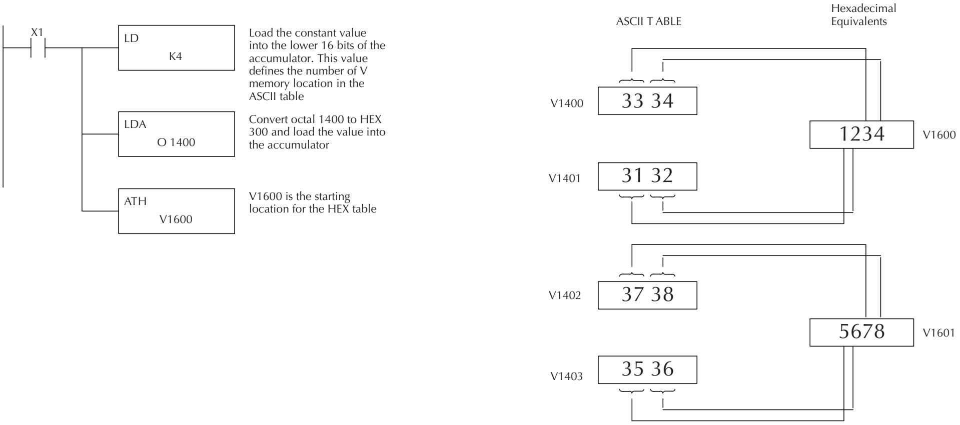
Step 3: Specify the starting V-memory location (Vaaa) for the HEX table in the ATH instruction.

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.

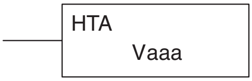
Operand Data Type		D4-454 Range
		aaa
V-memory	V	See memory map

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 Value	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


X1
LD
K4
Load the constant value into the lower 16 bits of the accumulator. This value defines the number of V memory location in the ASCII table
LDA
O 1400
Convert octal 1400 to HEX 300 and load the value into the accumulator
ATH
V1600
V1600 is the starting location for the HEX table
ASCII T ABLE
Hexadecimal Equivalents
V1400
33 34
1234
V1600
V1401
31 32
V1402
37 38
5678
V1601
V1403
35 36

HEX to ASCII (HTA)



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.

Step 1: Load the number of V-memory locations in the HEX table into the first level of the accumulator stack.

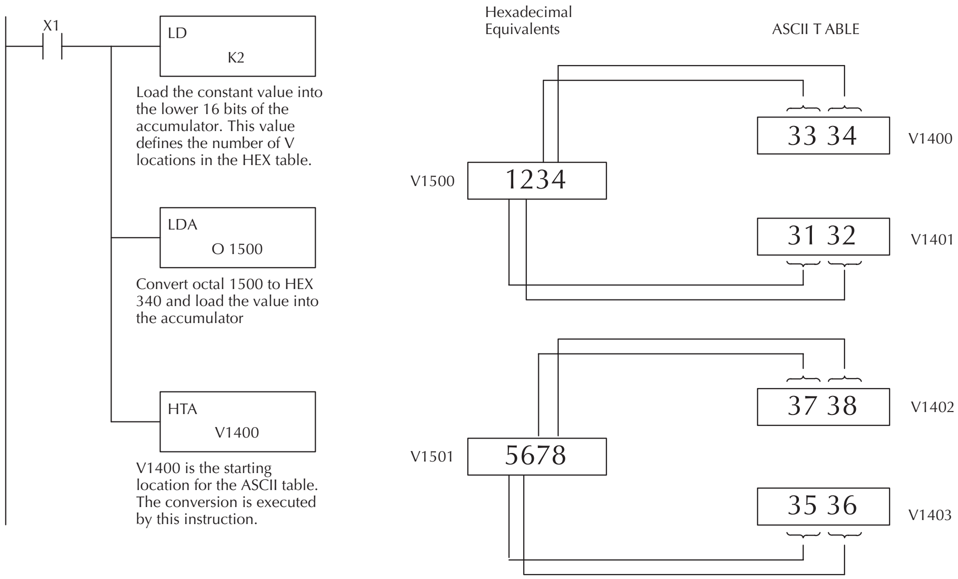
Step 2: Load the starting V-memory location for the HEX table into the accumulator. This parameter must be a HEX value.

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		D4-454 Range
		aaa
V-memory	V	See memory map

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.



Load the constant value into the lower 16 bits of the accumulator. This value defines the number of V locations in the HEX table.

Convert octal 1500 to HEX 340 and load the value into the accumulator

V1400 is the starting location for the ASCII table. The conversion is executed by this instruction.

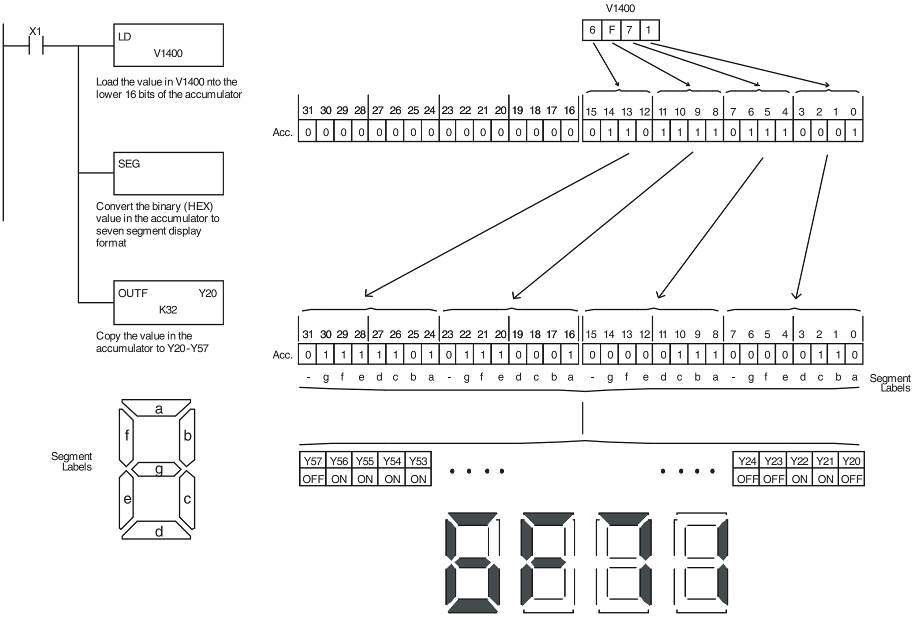
The table below lists valid ASCII values for HTA conversion.

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)

The BCD / Segment instruction converts a four digit HEX value in the accumulator to seven 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 HEX value in the accumulator is converted to seven segment format using the Segment instruction. The bit pattern in the accumulator is copied to Y20 – Y57 using the Out Formatted instruction.



Load the value in V1400 nto the lower 16 bits of the accumulator

Convert the binary (HEX) value in the accumulator to seven segment display format

Copy the value in the accumulator to Y20-Y57

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	1	1	0	0	0	1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	1	1	1	1	1	0	1	0	1	1	1	0	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0
Segment Labels	-	g	f	e	d	c	b	a	-	g	f	e	d	c	b	a	-	g	f	e	d	c	b	a	-	g	f	e	d	c	b	a

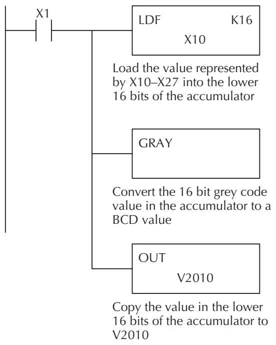
Y57	Y56	Y55	Y54	Y53
OFF	ON	ON	ON	ON

Y24	Y23	Y22	Y21	Y20
OFF	OFF	ON	ON	OFF

Gray Code (GRAY)

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.

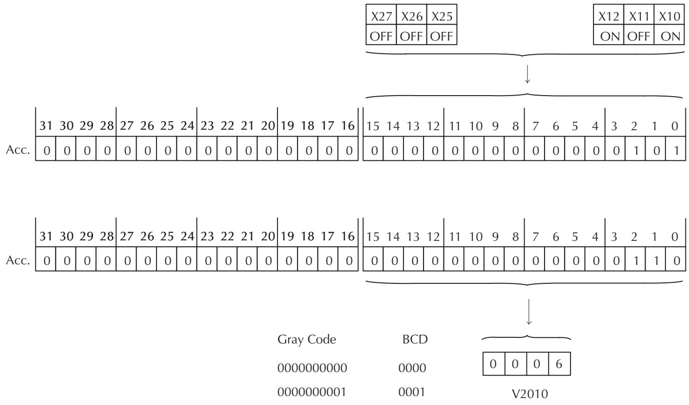
In the following example, when X1 is ON, the binary value represented by X10 – X27 is loaded into the accumulator using the Load Formatted instruction. The gray code value in the accumulator is converted to BCD using the Gray Code instruction. The value in the lower 16 bits of the accumulator is copied to V2010.



Load the value represented by X10–X27 into the lower 16 bits of the accumulator

Convert the 16 bit grey code value in the accumulator to a BCD value

Copy the value in the lower 16 bits of the accumulator to V2010



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Acc.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Gray Code	BCD
0000000000	0000
0000000001	0001
0000000011	0002
0000000010	0003
0000000110	0004
0000000111	0005
0000000101	0006
0000000100	0007
1000000001	1022
1000000000	1023

Shuffle Digits (SFLDGT)

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.

Step 1: Load the value (digits) to be shuffled into the first level of the accumulator stack.

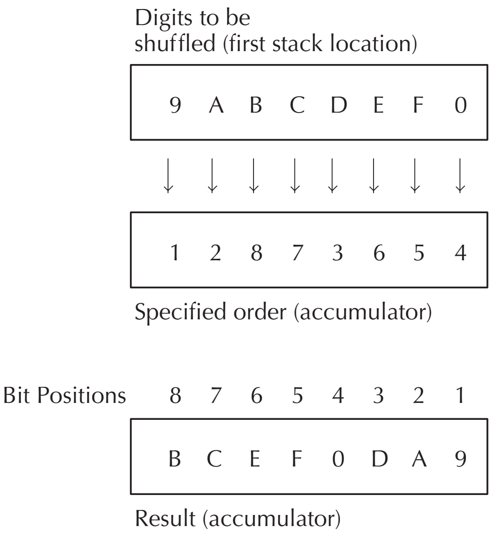
Step 2: Load the order that the digits will be shuffled to into the accumulator.

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.*

Shuffle Digits Block Diagram

There are a maximum of 8 digits that 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 the digits will be shuffled. The digits are shuffled and the result resides in the accumulator.


Digits to be shuffled (first stack location)

9 A B C D E F 0

1 2 8 7 3 6 5 4

Specified order (accumulator)

Bit Positions 8 7 6 5 4 3 2 1

B C E F 0 D A 9

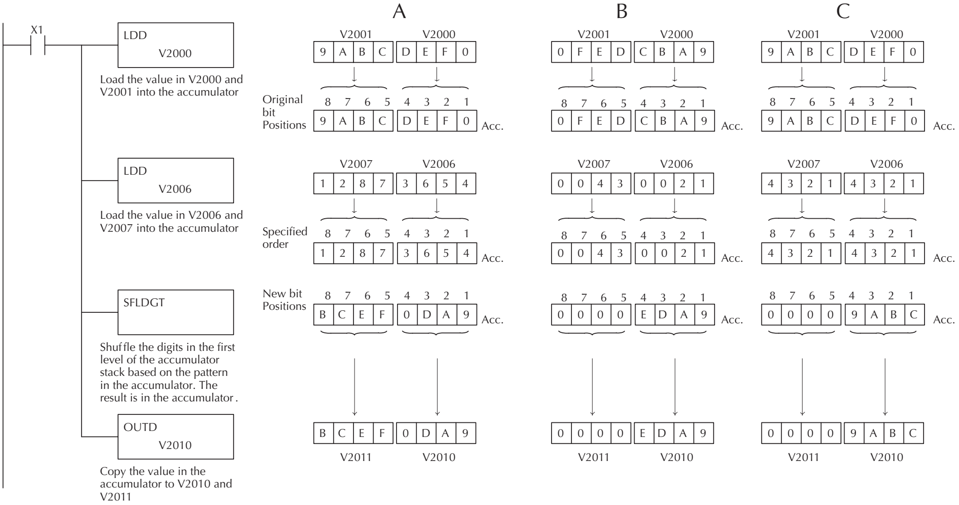
Result (accumulator)

In the following example, when 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.


X1
LDD V2000
LDD V2006
SFLDGT
OUTD V2010
A
B
C
Original bit Positions
Specified order
New bit Positions
Acc.
8 7 6 5 4 3 2 1
A: V2001 V2000 9 A B C D E F 0
A: V2007 V2006 1 2 8 7 3 6 5 4
A: New bit Positions B C E F 0 D A 9
A: V2011 V2010 B C E F 0 D A 9
B: V2001 V2000 0 F E D C B A 9
B: V2007 V2006 0 0 4 3 0 0 2 1
B: New bit Positions 0 0 0 0 E D A 9
B: V2011 V2010 0 0 0 0 E D A 9
C: V2001 V2000 9 A B C D E F 0
C: V2007 V2006 4 3 2 1 4 3 2 1
C: New bit Positions 0 0 0 0 9 A B C
C: V2011 V2010 0 0 0 0 9 A B C

Load the value in V2000 and V2001 into the accumulator

Load the value in V2006 and V2007 into the accumulator

Shuffle the digits in the first level of the accumulator stack based on the pattern in the accumulator. The result is in the accumulator.

Copy the value in the accumulator to V2010 and V2011

Table Instructions

Move (MOV)

The Move instruction moves the values from a V-memory table to another V-memory table the same length (a table being a consecutive group of V-memory locations). 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 MOV 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, 4096 decimal).

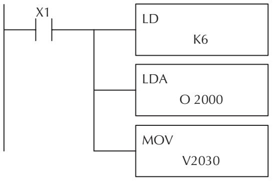
Step 2 Load the starting V-memory location for the locations to be moved into the accumulator. This parameter is a HEX value.

Step 3 Insert the MOV 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		D4-454 Range
		aaa
V-memory	V	See memory map
Pointer	P	See memory map

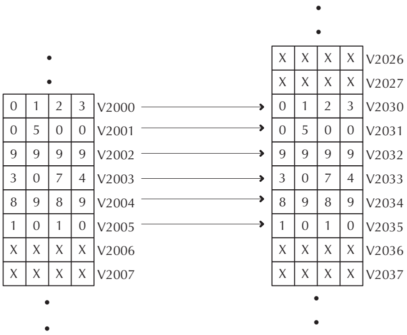
In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 2000 (V2000), the starting location for the source table, is loaded into the accumulator. The destination table location (V2030) is specified in the Move instruction.



Load the constant value 6 (HEX) into the lower 16 bits of the accumulator

Convert octal 2000 to HEX 400 and load the value into the accumulator

Copy the specified table locations to a table beginning at location V2030



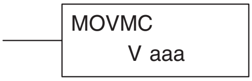
0	1	2	3	V2000
0	5	0	0	V2001
9	9	9	9	V2002
3	0	7	4	V2003
8	9	8	9	V2004
1	0	1	0	V2005
X	X	X	X	V2006
X	X	X	X	V2007

X	X	X	X	V2026
X	X	X	X	V2027
0	1	2	3	V2030
0	5	0	0	V2031
9	9	9	9	V2032
3	0	7	4	V2033
8	9	8	9	V2034
1	0	1	0	V2035
X	X	X	X	V2036
X	X	X	X	V2037

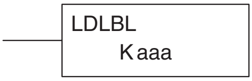
Move Memory Cartridge (MOVMC)

Load Label (LDLBL)

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 MOVMC and LDLBL 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 ladder memory and the beginning of the V-memory block into the first level of the 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 in V-memory (Vaaa).

This is the copy destination.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Constant	K	1 – FFFF

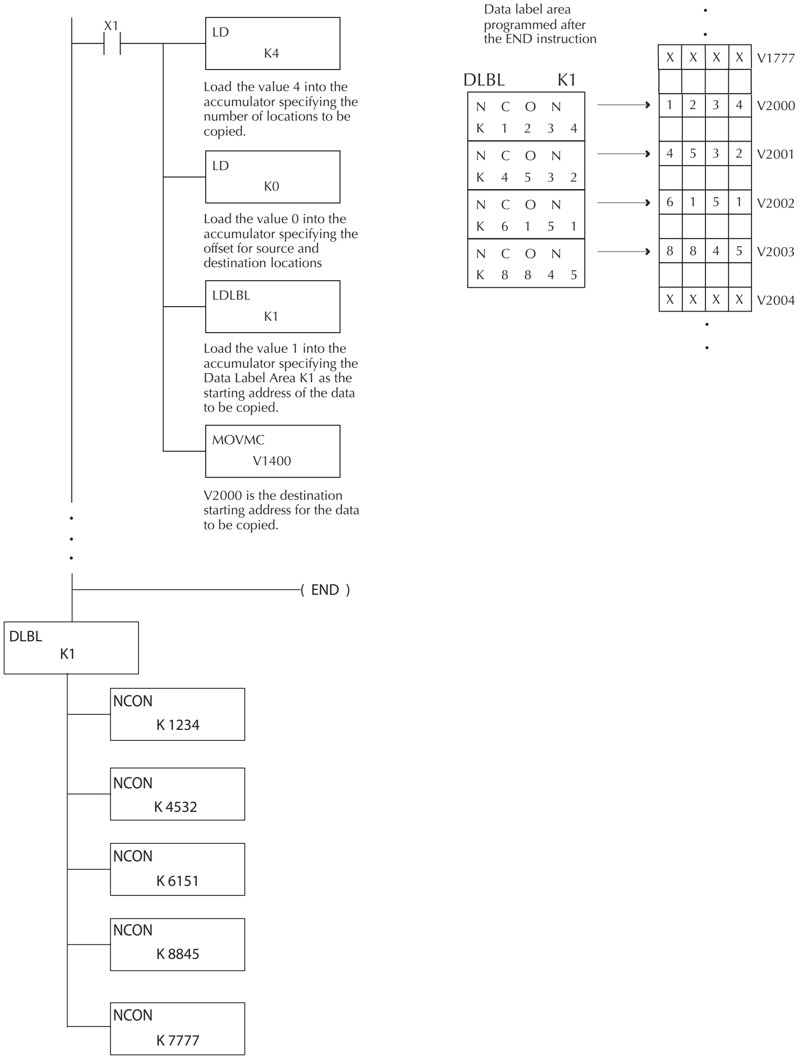
Discrete Bit Flags	Description
SP53	On if there is a table pointer error.

NOTE: *Status flags are only valid until:*
The end of the scan ; or another instruction that uses the same flag is executed.

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

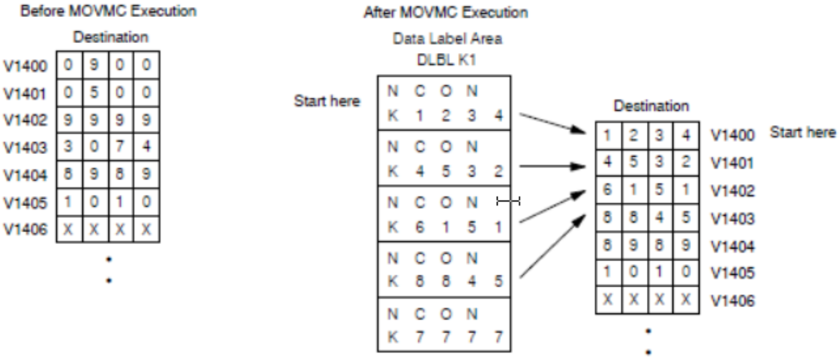
In the example below, data is copied from a Data Label Area to V-memory. When X1 is on, the constant value (K4) is loaded into the accumulator using the Load (LD) instruction. This value specifies the length of the table and is placed in the second stack location after the next Load and Load Label (LDLBL) instructions are executed. The constant value (K0) is loaded into the accumulator, specifying the offset for the source and destination data. It 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.


X1
LD
K4
Load the value 4 into the accumulator specifying the number of locations to be copied.
LD
K0
Load the value 0 into the accumulator specifying the offset for source and destination locations
LDLBL
K1
Load the value 1 into the accumulator specifying the Data Label Area K1 as the starting address of the data to be copied.
MOVMC
V1400
V2000 is the destination starting address for the data to be copied.
END
DLBL
K1
NCON
K 1234
NCON
K 4532
NCON
K 6151
NCON
K 8845
NCON
K 7777
Data label area programmed after the END instruction
DLBL K1
N C O N
K 1 2 3 4
N C O N
K 4 5 3 2
N C O N
K 6 1 5 1
N C O N
K 8 8 4 5
X X X X V1777
1 2 3 4 V2000
4 5 3 2 V2001
6 1 5 1 V2002
8 8 4 5 V2003
X X X X V2004

The following diagram shows the result of our example. The offset is equal to zero and four words will be copied in the Data Label area.

Example of Execution

Offset = 0, move 4 words



Before MOVMC Execution – Destination

V1400	0	9	0	0
V1401	0	5	0	0
V1402	9	9	9	9
V1403	3	0	7	4
V1404	8	9	8	9
V1405	1	0	1	0
V1406	X	X	X	X

After MOVMC Execution – Data Label Area DLBL K1

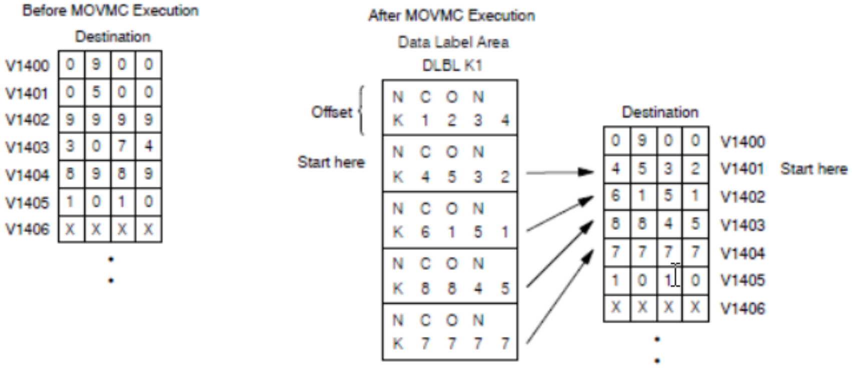
N	C	O	N	
K	1	2	3	4
N	C	O	N	
K	4	5	3	2
N	C	O	N	
K	6	1	5	1
N	C	O	N	
K	8	8	4	5
N	C	O	N	
K	7	7	7	7

Destination

1	2	3	4	V1400
4	5	3	2	V1401
6	1	5	1	V1402
8	8	4	5	V1403
8	9	8	9	V1404
1	0	1	0	V1405
X	X	X	X	V1406

The example is fairly straightforward when an offset of zero is used. However, it also helpful for you to understand the results that would have been obtained if different offset values (1 and 2) were used. Notice how the offset is used for both the data label (source) and the destination table. Also, notice how an improper offset (two in this case can result in unknown values being copied into the destination table.

Offset = 1, move 4 words



Before MOVMC Execution – Destination

V1400	0	9	0	0
V1401	0	5	0	0
V1402	9	9	9	9
V1403	3	0	7	4
V1404	8	9	8	9
V1405	1	0	1	0
V1406	X	X	X	X

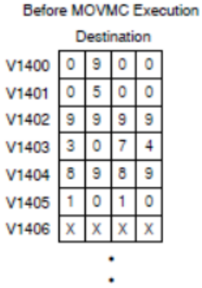
After MOVMC Execution – Data Label Area DLBL K1

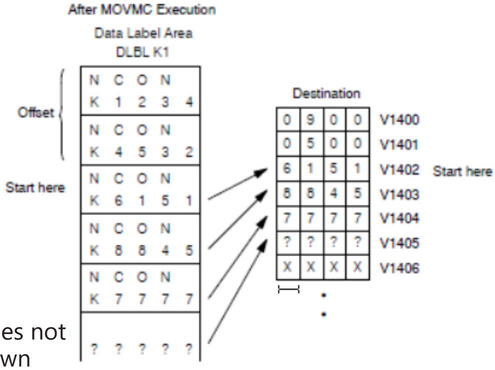
N	C	O	N	
K	1	2	3	4
N	C	O	N	
K	4	5	3	2
N	C	O	N	
K	6	1	5	1
N	C	O	N	
K	8	8	4	5
N	C	O	N	
K	7	7	7	7

Destination

0	9	0	0	V1400
4	5	3	2	V1401
6	1	5	1	V1402
8	8	4	5	V1403
7	7	7	7	V1404
1	0	1	0	V1405
X	X	X	X	V1406

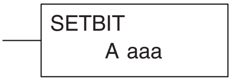
Offset = 2, move 4 words





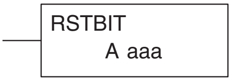
Since there is no NCON, the CPU does not know where to get the data. Unknown values will be copied into V1405.

SETBIT



The Set Bit instruction sets a single bit to one within a range of V-memory locations.

RSTBIT



The Reset Bit instruction resets a single bit to zero within a range of V-memory locations.

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".

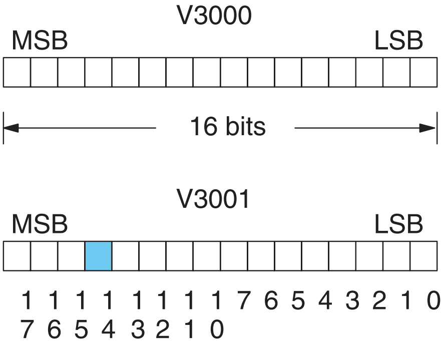
Helpful Hint: Remember that each V-memory location contains 16 bits. So, the bits of the first word of the table are numbered from 0 to 17 octal. For example, if the table length is six words, then 6 words = (6 x 16) bits, = 96 bits (decimal), or 140 octal. The permissible range of bit reference numbers would be 0 to 137 octal. SP 53 will be set if the bit specified is outside the range of the table.

Operand Data Type		D4-454 Range
		aaa
V-memory	V	See memory map

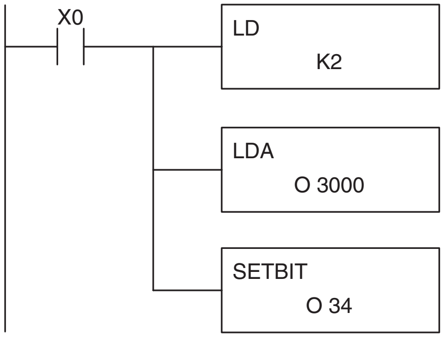
Discrete Bit Flags	Description
SP53	On when the specified bit is outside the range of the table.

***NOTE:** Status flags are only valid until the end of the scan or until 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 (2 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.



Load the constant value 2 (Hex.) into the lower 16 bits of the accumulator.

Convert octal 3000 to HEX and load the value into the accumulator. This is the table beginning.

Set bit 34 (octal) in the table to a ”1”.

Fill (FILL) 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 – FF.

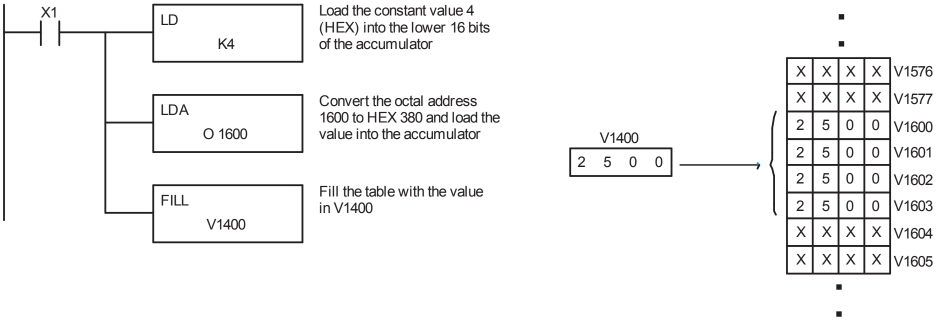
Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value.

Step 3: Insert the Fill instruction 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		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Constant	K	0 – FF

Discrete Bit Flags	Description
SP53	On if the 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.



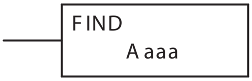
Load the constant value 4 (HEX) into the lower 16 bits of the accumulator

Convert the octal address 1600 to HEX 380 and load the value into the accumulator

Fill the table with the value in V1400

Find (FIND)

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 – FF.

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 (in HEX) is returned to the accumulator. 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.

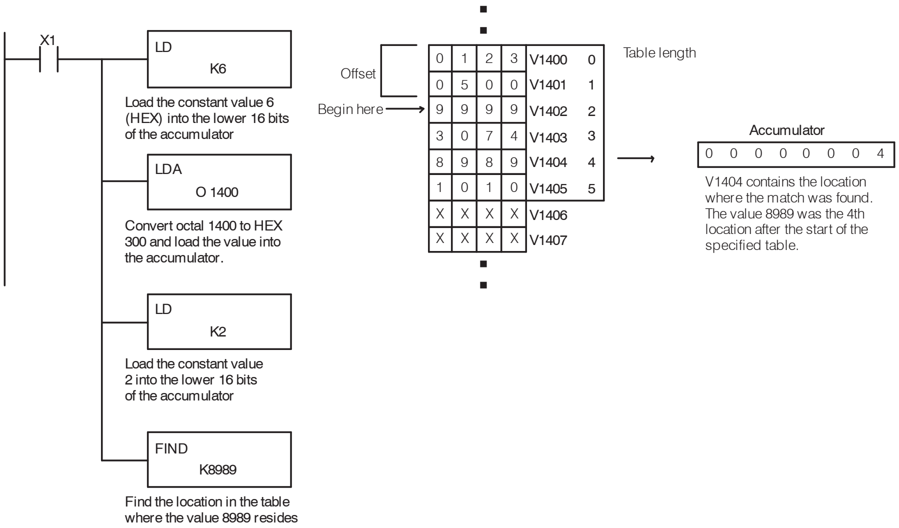
Helpful Hint: For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Constant	K	0 – FF

Discrete Bit Flags	Description
SP53	On if there is no value in the table that is equal to the search value.

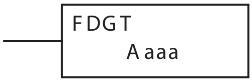
NOTE: *Status flags are only valid until another instruction that uses the same flags is executed. The pointer for this instruction starts at 0 and resides in the accumulator.*

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 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.


X1
LD
K6
Load the constant value 6 (HEX) into the lower 16 bits of the accumulator
LDA
O 1400
Convert octal 1400 to HEX 300 and load the value into the accumulator.
LD
K2
Load the constant value 2 into the lower 16 bits of the accumulator
FIND
K8989
Find the location in the table where the value 8989 resides
Offset
Begin here
Table length
Accumulator
0 0 0 0 0 0 0 4
V1404 contains the location where the match was found. The value 8989 was the 4th location after the start of the specified table.

0	1	2	3	V1400	0
0	5	0	0	V1401	1
9	9	9	9	V1402	2
3	0	7	4	V1403	3
8	9	8	9	V1404	4
1	0	1	0	V1405	5
X	X	X	X	V1406	
X	X	X	X	V1407	

Find Greater Than (FDGT)



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.

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 – FF.

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 (in HEX) which is returned to the accumulator. 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.

NOTE: *This instruction does not have an offset, such as the one required for the FIND instruction.*

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
Constant	K	0 – FF

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.

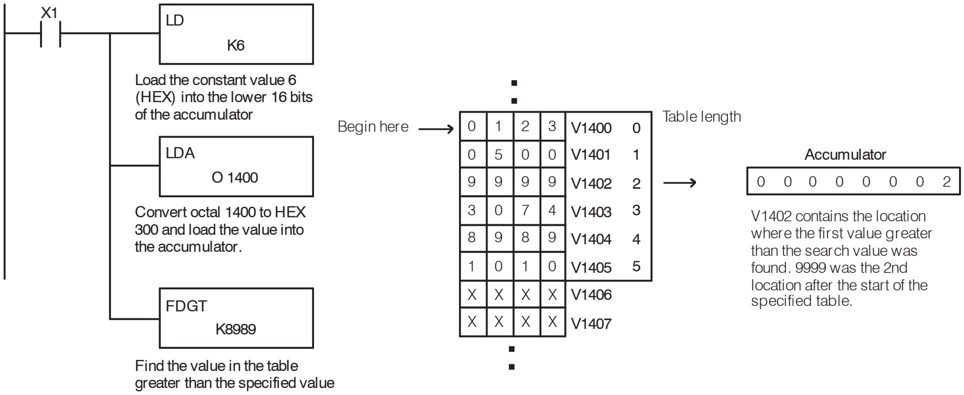

X1
LD
K6
Load the constant value 6 (HEX) into the lower 16 bits of the accumulator
LDA
O 1400
Convert octal 1400 to HEX 300 and load the value into the accumulator.
FDGT
K8989
Find the value in the table greater than the specified value
Begin here
0 1 2 3 V1400 0
0 5 0 0 V1401 1
9 9 9 9 V1402 2
3 0 7 4 V1403 3
8 9 8 9 V1404 4
1 0 1 0 V1405 5
X X X X V1406
X X X X V1407
Table length
Accumulator
0 0 0 0 0 0 0 2
V1402 contains the location where the first value greater than the search value was found. 9999 was the 2nd location after the start of the specified table.

Table to Destination (TTD)

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 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.

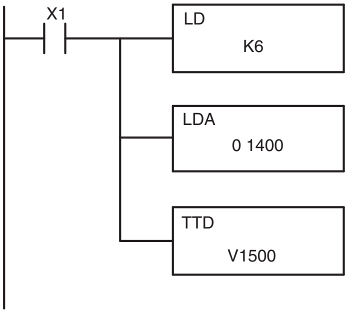
Helpful Hint: The pointer location should be set to the value where the table operation will begin. The special relay SP0 or a one-shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map

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 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.

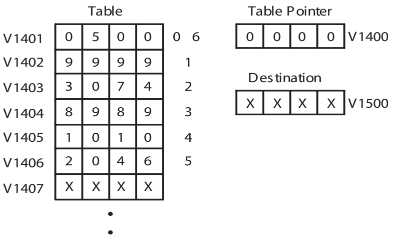


Load the constant value 6 (HEX) into the lower 16 bits of the accumulator

Convert octal 1400 to HEX 300 and load the value into the accumulator. This is the table pointer location

Copy the specified value from the table to the specified destination (V1500)

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 zero, and again when the pointer is equal to six. Why? Because the pointer is only equal to zero before the very first execution. From then on, it increments from one to six, and then resets to one.



Table

V1401	0	5	0	0	0 6
V1402	9	9	9	9	1
V1403	3	0	7	4	2
V1404	8	9	8	9	3
V1405	1	0	1	0	4
V1406	2	0	4	6	5
V1407	X	X	X	X	

Table Pointer

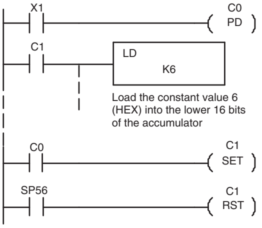
0	0	0	0	V1400

Destination

X	X	X	X	V1500

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.

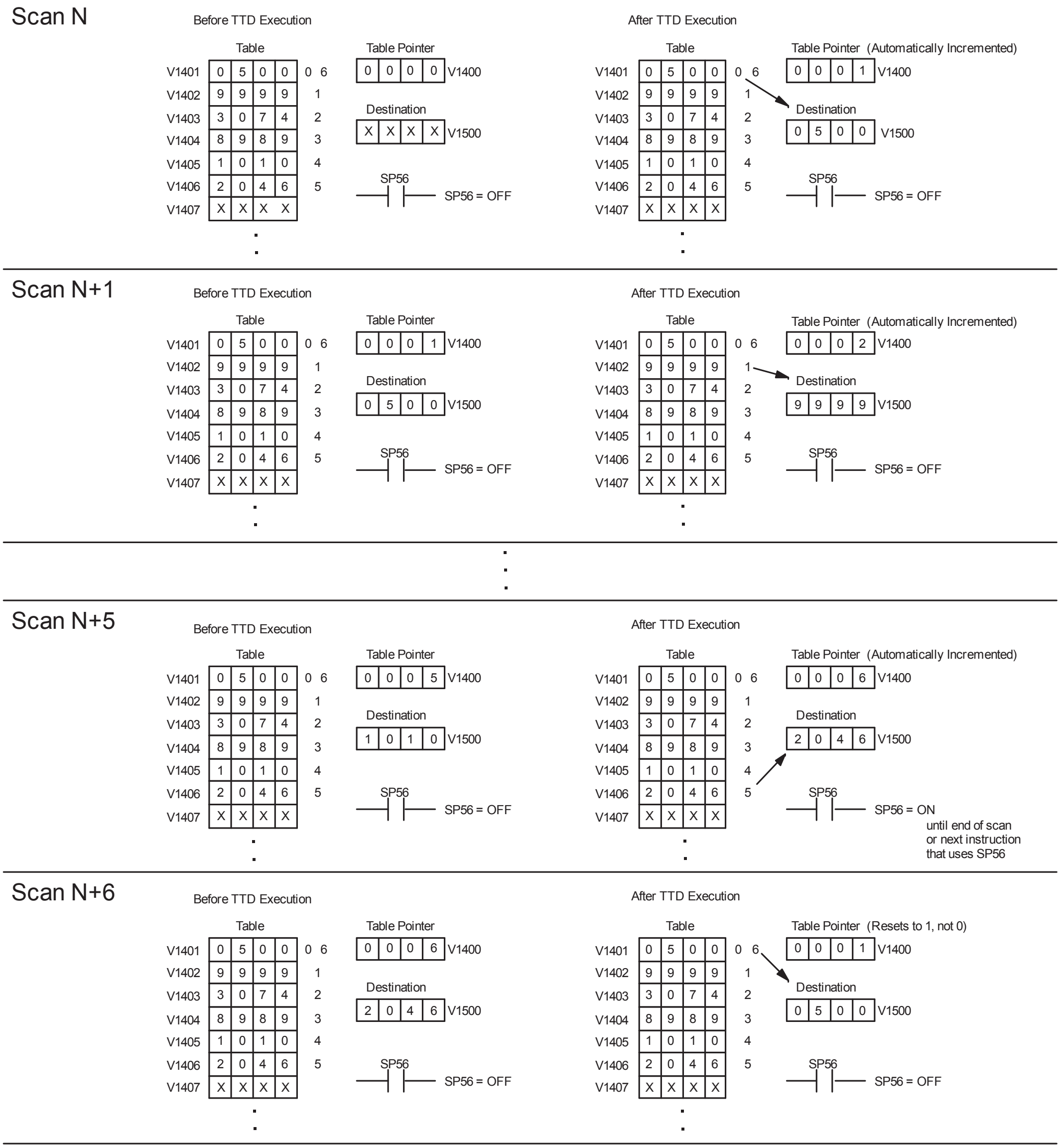
(optional latch example using SP56)



Load the constant value 6 (HEX) into the lower 16 bits of the accumulator

Since Special Relays are reset at the end of the scan, this latch must follow the TTD instruction in the program

The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically cycles from 0–6, 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)

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 two 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 instruction 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.

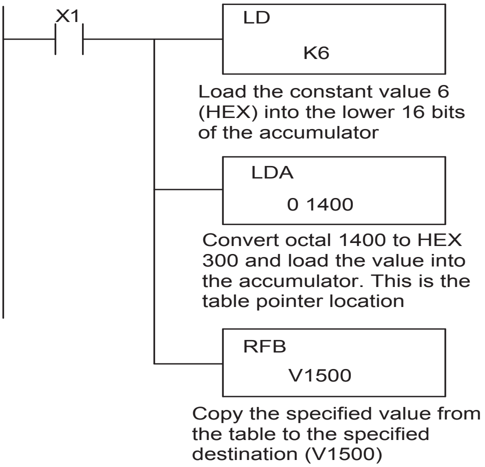
Helpful Hint: The pointer location should be set to the value where the table operation will begin. The special relay SP0 or a one-shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map

Discrete Bit Flags	Description
SP56	On when the table pointer equals zero.

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 must 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.



Load the constant value 6 (HEX) into the lower 16 bits of the accumulator

Convert octal 1400 to HEX 300 and load the value into the accumulator. This is the table pointer location

Copy the specified value from the table to the specified destination (V1500)

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 one. The second data location, V1402, will be used when the pointer is equal to two, etc.

Table

V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

•
•

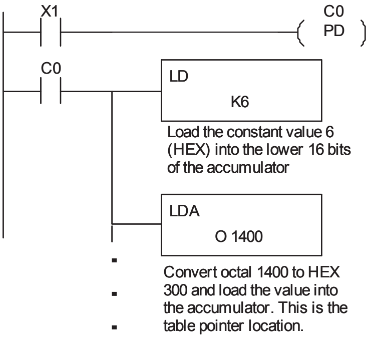
Table Pointer

0	0	0	0	V1400

Destination

X	X	X	X	V1500

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 table would cycle through the locations very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to remove one value each time the input contact transitions from low to high.

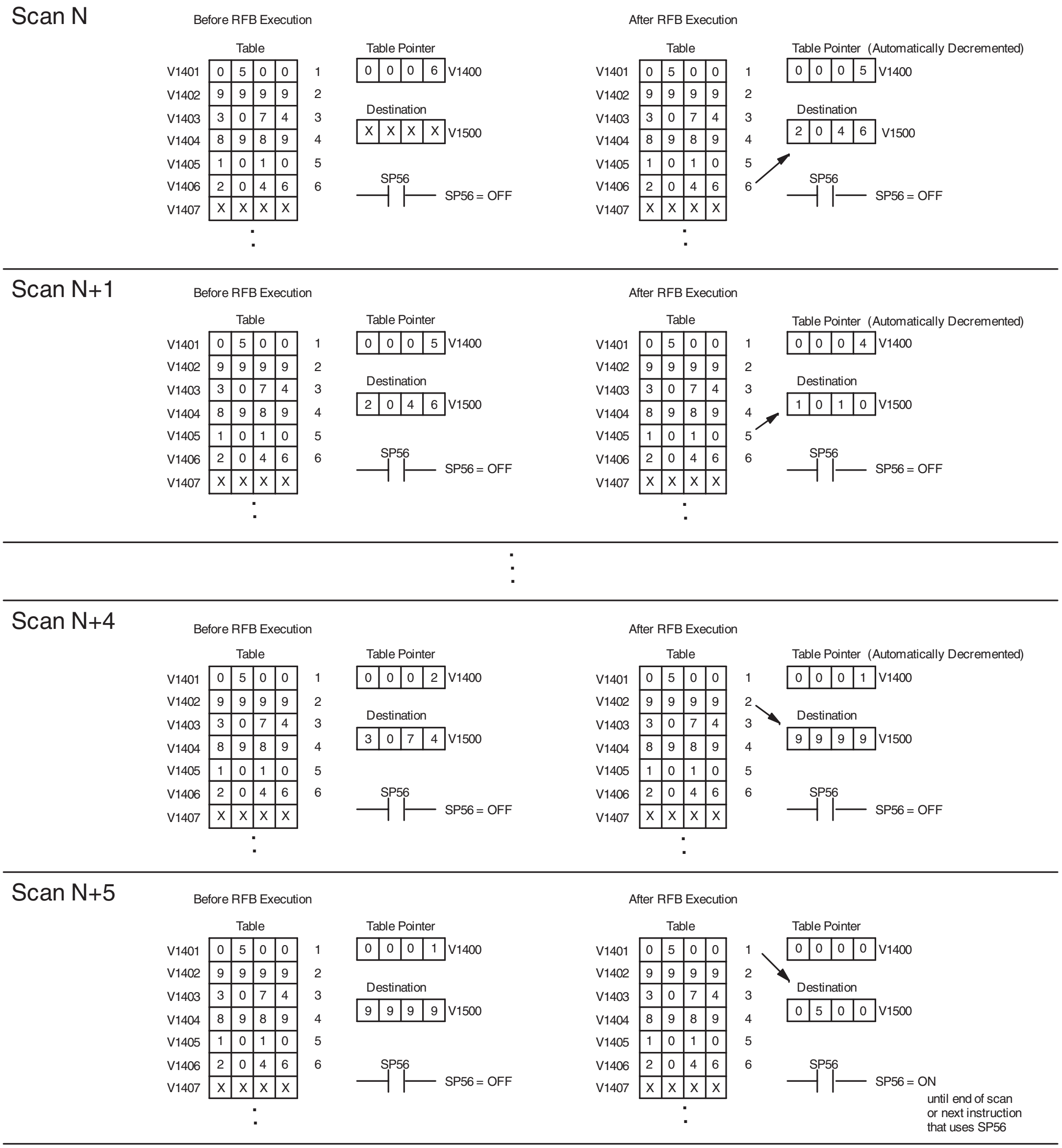


Load the constant value 6 (HEX) into the lower 16 bits of the accumulator

Convert octal 1400 to HEX 300 and load the value into the accumulator. This is the table pointer location.

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

Before RFB Execution

	Table				
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Pointer: 0 0 0 6 V1400

Destination: X X X X V1500

SP56 SP56 = OFF

After RFB Execution

	Table				
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Pointer (Automatically Decremented): 0 0 0 5 V1400

Destination: 2 0 4 6 V1500

SP56 SP56 = OFF

Scan N+1

Before RFB Execution

	Table				
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Pointer: 0 0 0 5 V1400

Destination: 2 0 4 6 V1500

SP56 SP56 = OFF

After RFB Execution

	Table				
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Pointer (Automatically Decremented): 0 0 0 4 V1400

Destination: 1 0 1 0 V1500

SP56 SP56 = OFF

⋮

Scan N+4

Before RFB Execution

	Table				
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Pointer: 0 0 0 2 V1400

Destination: 3 0 7 4 V1500

SP56 SP56 = OFF

After RFB Execution

	Table				
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Pointer (Automatically Decremented): 0 0 0 1 V1400

Destination: 9 9 9 9 V1500

SP56 SP56 = OFF

Scan N+5

Before RFB Execution

	Table				
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Pointer: 0 0 0 1 V1400

Destination: 9 9 9 9 V1500

SP56 SP56 = OFF

After RFB Execution

	Table				
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Pointer: 0 0 0 0 V1400

Destination: 0 5 0 0 V1500

SP56 SP56 = ON

until end of scan
or next instruction
that uses SP56

Source to Table (STT)

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.

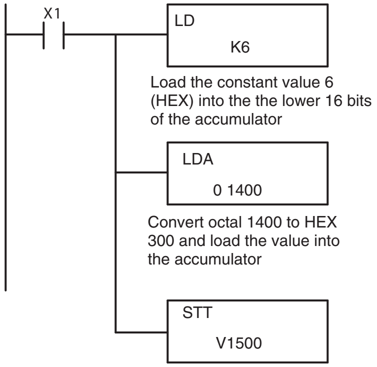
Helpful Hint: The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the pointer should be between 0 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		D4-454 Range
	A	aaa
V-memory	V	See memory map

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.

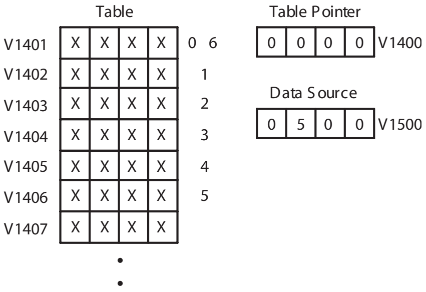


Load the constant value 6 (HEX) into the the lower 16 bits of the accumulator

Convert octal 1400 to HEX 300 and load the value into the accumulator

Copy the specified value from the source location (V1500) to the table

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 zero, and again when the pointer is equal to six. Why? Because the pointer is only equal to zero before the very first execution. From then on, it increments from one to six, and then resets to one.



Table

V1401	X	X	X	X	0 6
V1402	X	X	X	X	1
V1403	X	X	X	X	2
V1404	X	X	X	X	3
V1405	X	X	X	X	4
V1406	X	X	X	X	5
V1407	X	X	X	X	

Table Pointer

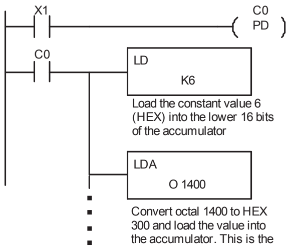
0	0	0	0	V1400

Data Source

0	5	0	0	V1500

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 one value each time the input contact transitions from low to high.

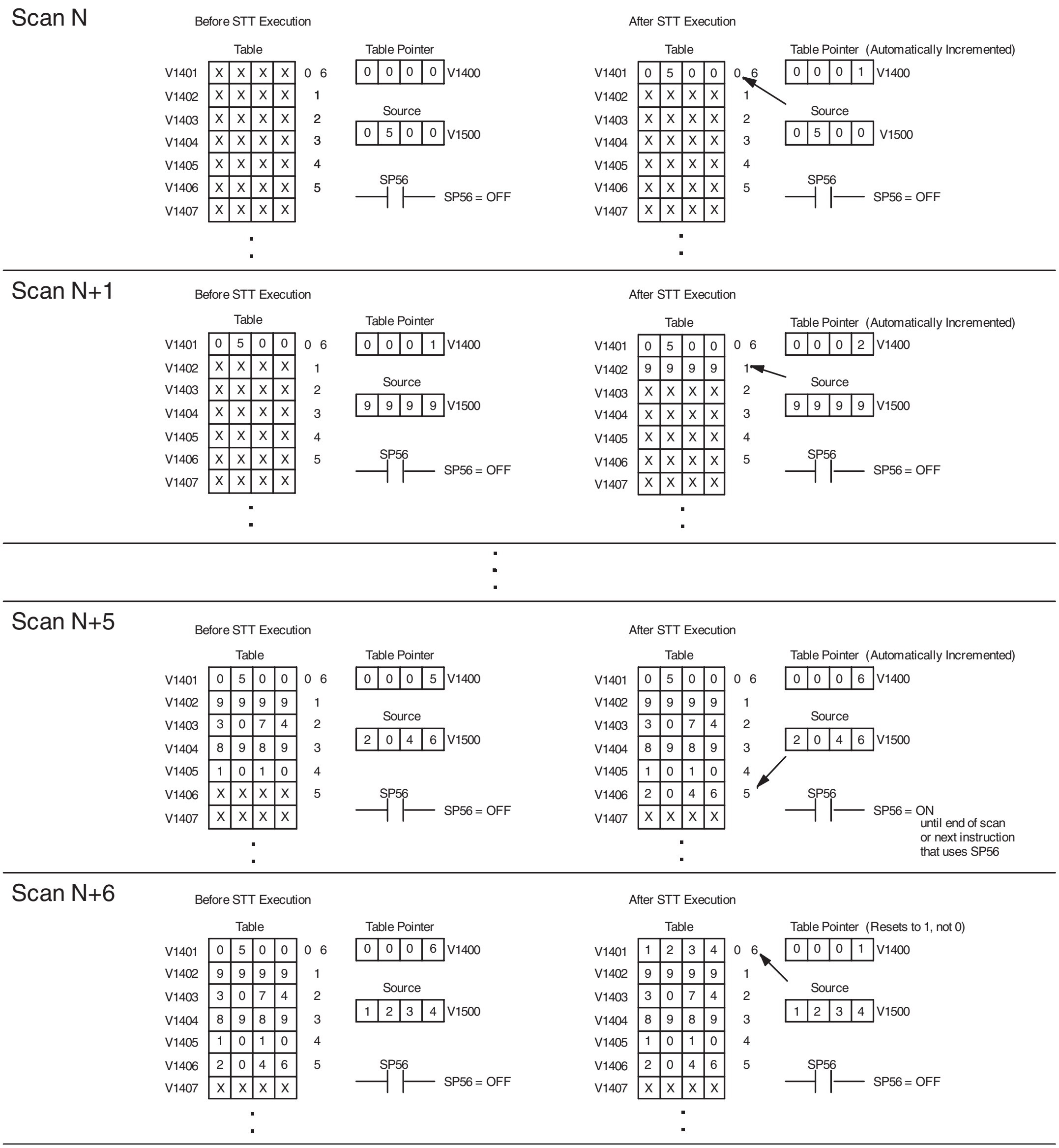
(optional one-shot method)



Load the constant value 6 (HEX) into the lower 16 bits of the accumulator

Convert octal 1400 to HEX 300 and load the value into the accumulator. This is the starting table location.

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.



Scan N

Before STT Execution

	Table				
V1401	X	X	X	X	0 6
V1402	X	X	X	X	1
V1403	X	X	X	X	2
V1404	X	X	X	X	3
V1405	X	X	X	X	4
V1406	X	X	X	X	5
V1407	X	X	X	X	

Table Pointer: 0 0 0 0 V1400

Source: 0 5 0 0 V1500

SP56 = OFF

After STT Execution

	Table				
V1401	0	5	0	0	0 6
V1402	X	X	X	X	1
V1403	X	X	X	X	2
V1404	X	X	X	X	3
V1405	X	X	X	X	4
V1406	X	X	X	X	5
V1407	X	X	X	X	

Table Pointer (Automatically Incremented): 0 0 0 1 V1400

Source: 0 5 0 0 V1500

SP56 = OFF

Scan N+1

Before STT Execution

	Table				
V1401	0	5	0	0	0 6
V1402	X	X	X	X	1
V1403	X	X	X	X	2
V1404	X	X	X	X	3
V1405	X	X	X	X	4
V1406	X	X	X	X	5
V1407	X	X	X	X	

Table Pointer: 0 0 0 1 V1400

Source: 9 9 9 9 V1500

SP56 = OFF

After STT Execution

	Table				
V1401	0	5	0	0	0 6
V1402	9	9	9	9	1
V1403	X	X	X	X	2
V1404	X	X	X	X	3
V1405	X	X	X	X	4
V1406	X	X	X	X	5
V1407	X	X	X	X	

Table Pointer (Automatically Incremented): 0 0 0 2 V1400

Source: 9 9 9 9 V1500

SP56 = OFF

Scan N+5

Before STT Execution

	Table				
V1401	0	5	0	0	0 6
V1402	9	9	9	9	1
V1403	3	0	7	4	2
V1404	8	9	8	9	3
V1405	1	0	1	0	4
V1406	X	X	X	X	5
V1407	X	X	X	X	

Table Pointer: 0 0 0 5 V1400

Source: 2 0 4 6 V1500

SP56 = OFF

After STT Execution

	Table				
V1401	0	5	0	0	0 6
V1402	9	9	9	9	1
V1403	3	0	7	4	2
V1404	8	9	8	9	3
V1405	1	0	1	0	4
V1406	2	0	4	6	5
V1407	X	X	X	X	

Table Pointer (Automatically Incremented): 0 0 0 6 V1400

Source: 2 0 4 6 V1500

SP56 = ON until end of scan or next instruction that uses SP56

Scan N+6

Before STT Execution

	Table				
V1401	0	5	0	0	0 6
V1402	9	9	9	9	1
V1403	3	0	7	4	2
V1404	8	9	8	9	3
V1405	1	0	1	0	4
V1406	2	0	4	6	5
V1407	X	X	X	X	

Table Pointer: 0 0 0 6 V1400

Source: 1 2 3 4 V1500

SP56 = OFF

After STT Execution

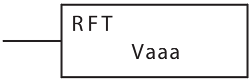
	Table				
V1401	1	2	3	4	0 6
V1402	9	9	9	9	1
V1403	3	0	7	4	2
V1404	8	9	8	9	3
V1405	1	0	1	0	4
V1406	2	0	4	6	5
V1407	X	X	X	X	

Table Pointer (Resets to 1, not 0): 0 0 0 1 V1400

Source: 1 2 3 4 V1500

SP56 = OFF

Remove from Table (RFT)



The Remove From Table instruction pops a value off 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 zero 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 two 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.

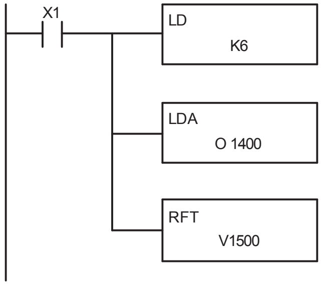
Helpful Hint: The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the table counter should be between 1 and 6. If the value is outside of this range or zero, 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		D4-454 Range
	A	aaa
V-memory	V	See memory map

Discrete Bit Flags	Description
SP56	On when the table pointer equals zero.

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 must 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.



Load the constant value 6 (Hex.) into the lower 16 bits of the accumulator

Convert octal 1400 to HEX 300 and load the value into the accumulator

Copy the specified value from the table to the specified location (V1500)

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 six of the locations would be affected during the instruction execution.

Table

V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

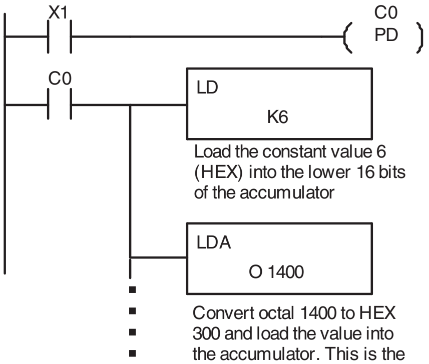
Table Counter

0	0	0	6	V1400

Destination

X	X	X	X	V1500

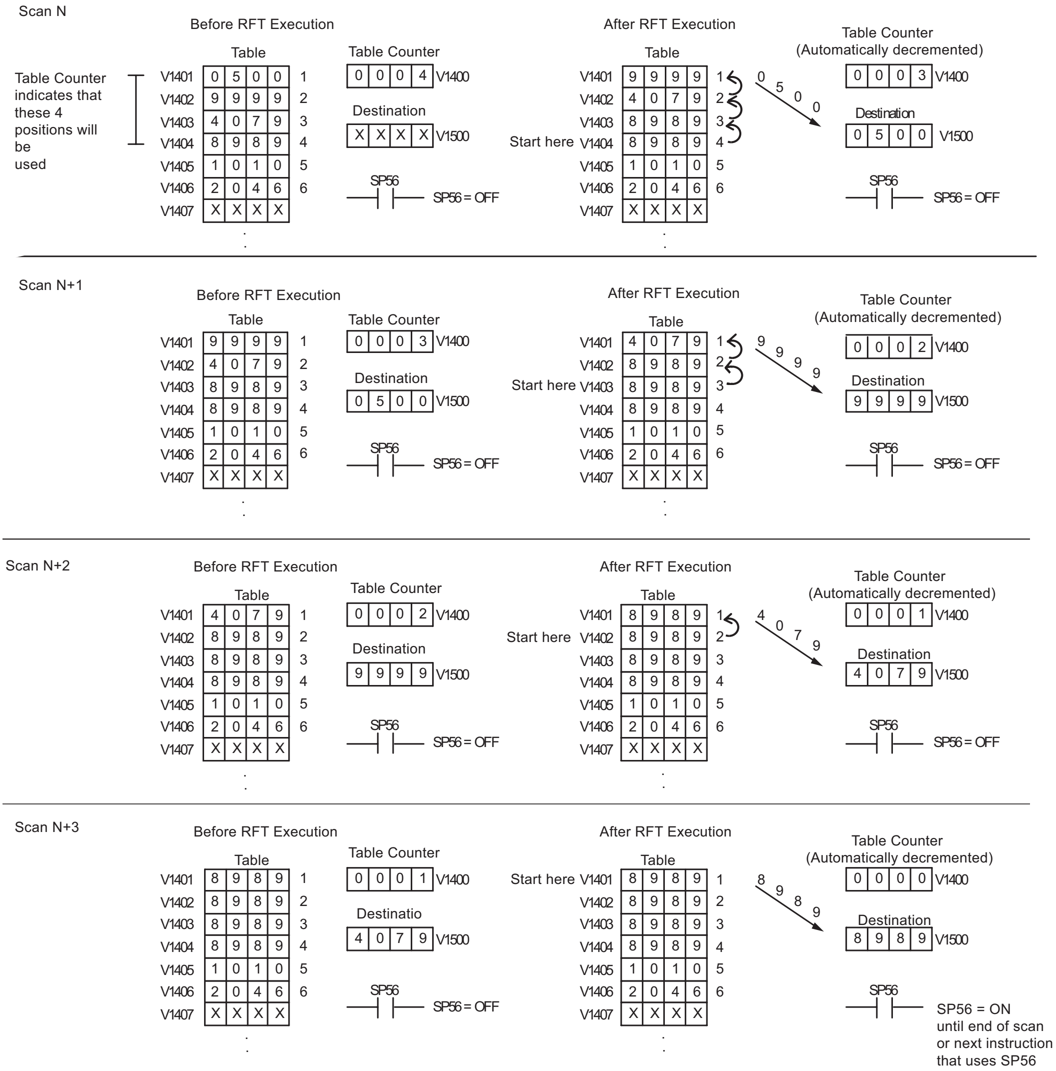
Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer decrements automatically, the data would be removed from the table very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to remove one value each time the input contact transitions from low to high.



Load the constant value 6 (HEX) into the lower 16 bits of the accumulator

Convert octal 1400 to HEX 300 and load the value into the accumulator. This is the table pointer location.

The following diagram shows the scan-by-scan results of the execution for our example program. In our example, we show 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 two table positions, 5 and 6, are not moved up through the table. Also, notice that SP56, which comes on when the table counter is zero, is only on until the end of the scan.



Scan N

Table Counter indicates that these 4 positions will be used

Before RFT Execution

Table					
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	4	0	7	9	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Counter: 0 0 0 4 V1400

Destination: X X X X V1500

SP56 — SP56 = OFF

After RFT Execution

Table					
V1401	9	9	9	9	1
V1402	4	0	7	9	2
V1403	8	9	8	9	3
Start here V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

0 5 0 0

Table Counter (Automatically decremented): 0 0 0 3 V1400

Destination: 0 5 0 0 V1500

SP56 — SP56 = OFF

Scan N+1

Before RFT Execution

Table					
V1401	9	9	9	9	1
V1402	4	0	7	9	2
V1403	8	9	8	9	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Counter: 0 0 0 3 V1400

Destination: 0 5 0 0 V1500

SP56 — SP56 = OFF

After RFT Execution

Table					
V1401	4	0	7	9	1
V1402	8	9	8	9	2
Start here V1403	8	9	8	9	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

9 9 9 9

Table Counter (Automatically decremented): 0 0 0 2 V1400

Destination: 9 9 9 9 V1500

SP56 — SP56 = OFF

Scan N+2

Before RFT Execution

Table					
V1401	4	0	7	9	1
V1402	8	9	8	9	2
V1403	8	9	8	9	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Counter: 0 0 0 2 V1400

Destination: 9 9 9 9 V1500

SP56 — SP56 = OFF

After RFT Execution

Table					
V1401	8	9	8	9	1
Start here V1402	8	9	8	9	2
V1403	8	9	8	9	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

4 0 7 9

Table Counter (Automatically decremented): 0 0 0 1 V1400

Destination: 4 0 7 9 V1500

SP56 — SP56 = OFF

Scan N+3

Before RFT Execution

Table					
V1401	8	9	8	9	1
V1402	8	9	8	9	2
V1403	8	9	8	9	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Counter: 0 0 0 1 V1400

Destinatio: 4 0 7 9 V1500

SP56 — SP56 = OFF

After RFT Execution

Table					
Start here V1401	8	9	8	9	1
V1402	8	9	8	9	2
V1403	8	9	8	9	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

8 9 8 9

Table Counter (Automatically decremented): 0 0 0 0 V1400

Destination: 8 9 8 9 V1500

SP56 — SP56 = ON until end of scan or next instruction that uses SP56

Add to Top (ATT)

The Add To Top instruction pushes a value on to 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 two additional instructions. Listed below are the steps necessary to program the Add To Top 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 ATT instructions which specifies source V-memory location (Vaaa). This is where the value will be moved from.

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: 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.

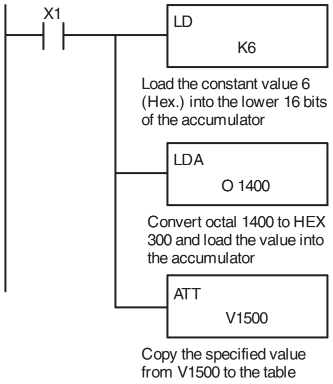
Helpful Hint: The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the table counter should be between 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		D4-454 Range
	A	aaa
V-memory	V	See memory map

Discrete Bit Flags	Description
SP56	On when the table pointer equal to the table size.

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 must 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.



Load the constant value 6 (Hex.) into the lower 16 bits of the accumulator

Convert octal 1400 to HEX 300 and load the value into the accumulator

Copy the specified value from V1500 to the table

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 easily be calculated 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.

Table

V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

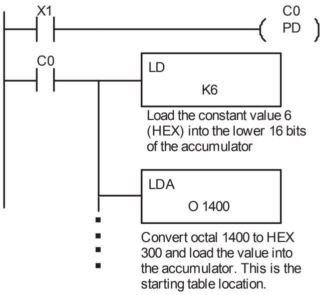
Table Counter

0	0	0	2	V1400

Data Source

X	X	X	X	V1500

(e.g.: 6 - 2 = 4)

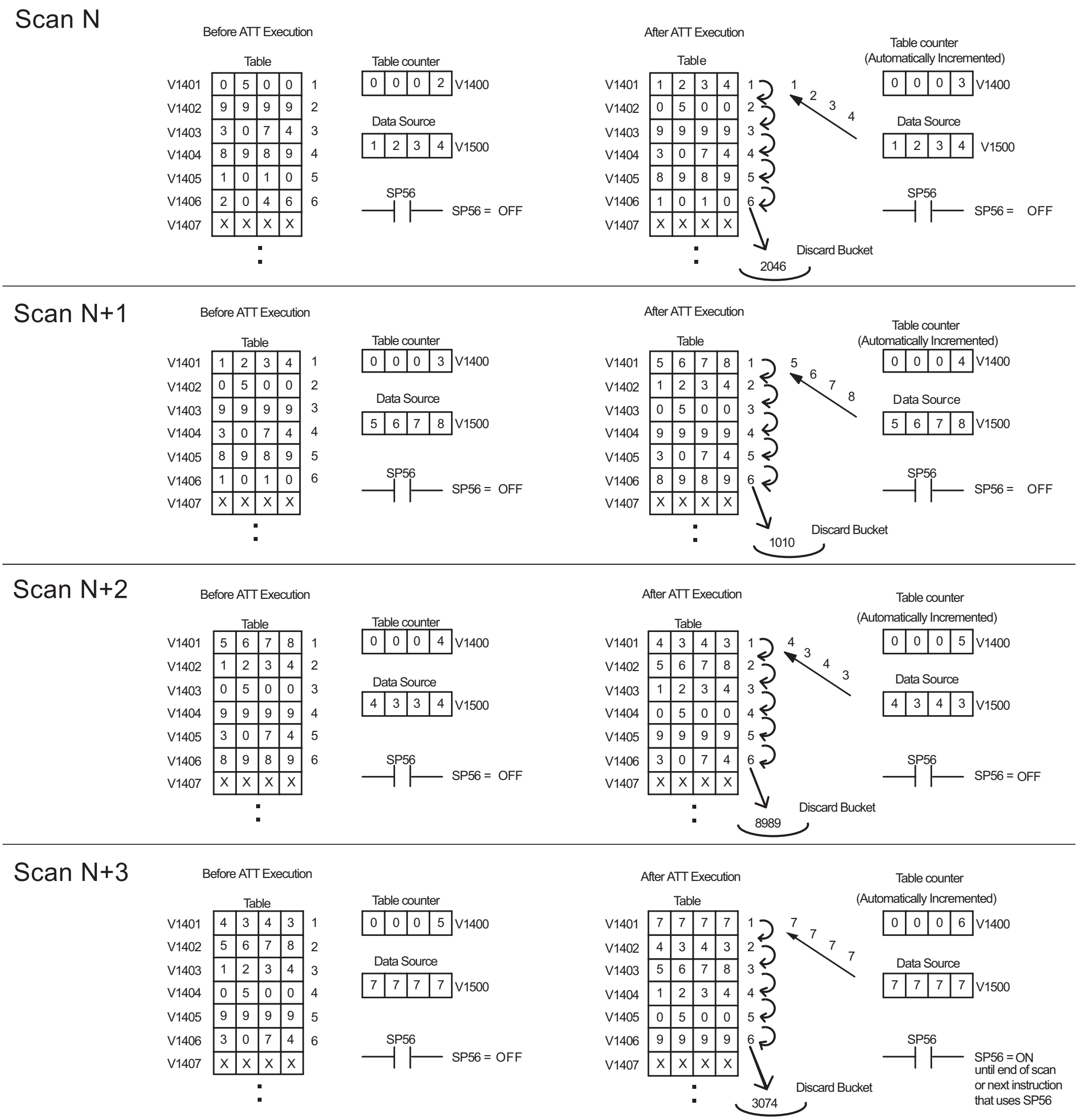


Load the constant value 6 (HEX) into the lower 16 bits of the accumulator

Convert octal 1400 to HEX 300 and load the value into the accumulator. This is the starting table location.

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

	Before ATT Execution Table			After ATT Execution Table	
V1401	0 5 0 0	1	V1401	1 2 3 4	1
V1402	9 9 9 9	2	V1402	0 5 0 0	2
V1403	3 0 7 4	3	V1403	9 9 9 9	3
V1404	8 9 8 9	4	V1404	3 0 7 4	4
V1405	1 0 1 0	5	V1405	8 9 8 9	5
V1406	2 0 4 6	6	V1406	1 0 1 0	6
V1407	X X X X		V1407	X X X X	

Before ATT Execution
Table counter 0 0 0 2 V1400
Data Source 1 2 3 4 V1500
SP56
SP56 = OFF
After ATT Execution
1 2 3 4
Table counter (Automatically Incremented) 0 0 0 3 V1400
Data Source 1 2 3 4 V1500
SP56
SP56 = OFF
Discard Bucket
2046

Scan N+1

	Before ATT Execution Table			After ATT Execution Table	
V1401	1 2 3 4	1	V1401	5 6 7 8	1
V1402	0 5 0 0	2	V1402	1 2 3 4	2
V1403	9 9 9 9	3	V1403	0 5 0 0	3
V1404	3 0 7 4	4	V1404	9 9 9 9	4
V1405	8 9 8 9	5	V1405	3 0 7 4	5
V1406	1 0 1 0	6	V1406	8 9 8 9	6
V1407	X X X X		V1407	X X X X	

Before ATT Execution
Table counter 0 0 0 3 V1400
Data Source 5 6 7 8 V1500
SP56
SP56 = OFF
After ATT Execution
5 6 7 8
Table counter (Automatically Incremented) 0 0 0 4 V1400
Data Source 5 6 7 8 V1500
SP56
SP56 = OFF
Discard Bucket
1010

Scan N+2

	Before ATT Execution Table			After ATT Execution Table	
V1401	5 6 7 8	1	V1401	4 3 4 3	1
V1402	1 2 3 4	2	V1402	5 6 7 8	2
V1403	0 5 0 0	3	V1403	1 2 3 4	3
V1404	9 9 9 9	4	V1404	0 5 0 0	4
V1405	3 0 7 4	5	V1405	9 9 9 9	5
V1406	8 9 8 9	6	V1406	3 0 7 4	6
V1407	X X X X		V1407	X X X X	

Before ATT Execution
Table counter 0 0 0 4 V1400
Data Source 4 3 3 4 V1500
SP56
SP56 = OFF
After ATT Execution
4 3 4 3
Table counter (Automatically Incremented) 0 0 0 5 V1400
Data Source 4 3 4 3 V1500
SP56
SP56 = OFF
Discard Bucket
8989

Scan N+3

	Before ATT Execution Table			After ATT Execution Table	
V1401	4 3 4 3	1	V1401	7 7 7 7	1
V1402	5 6 7 8	2	V1402	4 3 4 3	2
V1403	1 2 3 4	3	V1403	5 6 7 8	3
V1404	0 5 0 0	4	V1404	1 2 3 4	4
V1405	9 9 9 9	5	V1405	0 5 0 0	5
V1406	3 0 7 4	6	V1406	9 9 9 9	6
V1407	X X X X		V1407	X X X X	

Before ATT Execution
Table counter 0 0 0 5 V1400
Data Source 7 7 7 7 V1500
SP56
SP56 = OFF
After ATT Execution
7 7 7 7
Table counter (Automatically Incremented) 0 0 0 6 V1400
Data Source 7 7 7 7 V1500
SP56
SP56 = ON until end of scan or next instruction that uses SP56
Discard Bucket
3074

Table Shift Left (TSHFL)

The Table Shift Left instruction shifts all the bits in a V-memory table to the left, the specified number of bit positions.

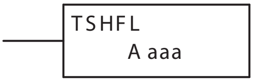
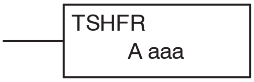
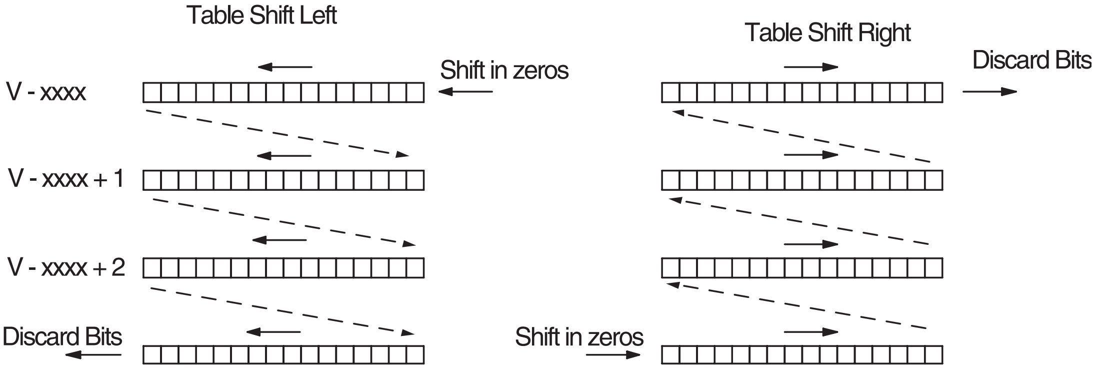


Table Shift Right (TSHFR)

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 just 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.

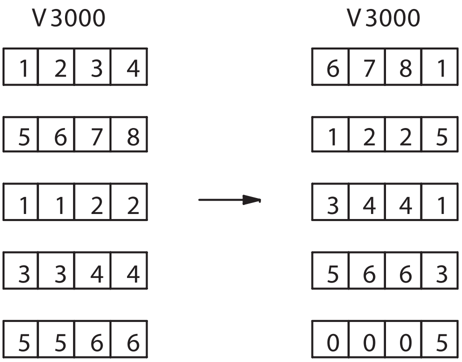
Helpful hint: Remember that each V-memory location contains 16 bits. So, the bits of the first word of the table are numbered from 0 to 17 octal. If you want to shift the entire table by 20 bits, that is 24 octal. SP 53 will be set if the number of bits to be shifted is larger than the total bits contained within the table. SP 67 will be set if the last bit shifted (just before it is discarded) is a "1".

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map

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

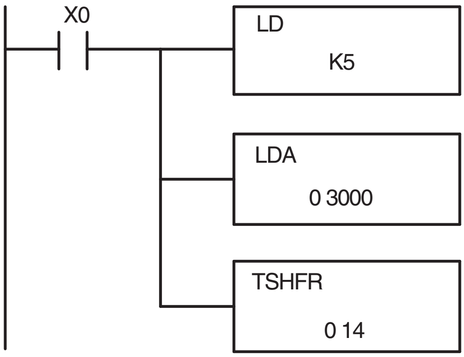
NOTE: *Status flags are only valid until the end of the scan or another instruction that uses the same flag is executed.*

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.



V3000		V3000
1 2 3 4		6 7 8 1
5 6 7 8		1 2 2 5
1 1 2 2	→	3 4 4 1
3 3 4 4		5 6 6 3
5 5 6 6		0 0 0 5

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.



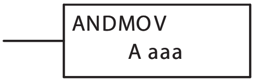
Load the constant value 5 (Hex.) into the lower 16 bits of the accumulator.

Convert octal 3000 to HEX and load the value into the accumulator. This is the table beginning.

Do a table shift right by 12 bits, which is 14 octal.

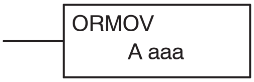
AND Move (ANDMOV)

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.



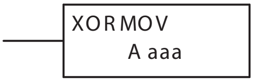
OR Move (ORMOV)

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)

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.

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.

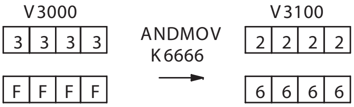
Step 2: Load the starting V-memory location for the table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

Step 3: Load the BCD/hex bit pattern into the accumulator which will be logically combined with the table contents as they are copied.

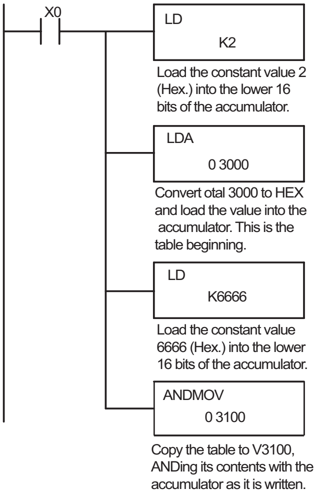
Step 4: Insert the AND Move, OR Move, or XOR Move instruction. This specifies the starting location of the copy of the original table. This new table will automatically be the same length as the original table.

Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map

The example table below 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 to the right 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.



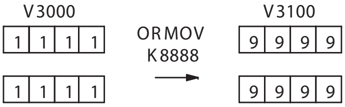
Load the constant value 2 (Hex.) into the lower 16 bits of the accumulator.

Convert otal 3000 to HEX and load the value into the accumulator. This is the table beginning.

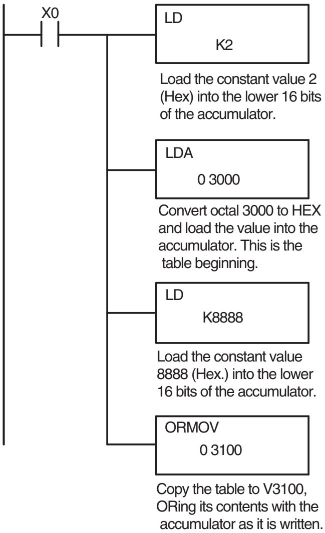
Load the constant value 6666 (Hex.) into the lower 16 bits of the accumulator.

Copy the table to V3100, ANDing its contents with the accumulator as it is written.

The example below 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.



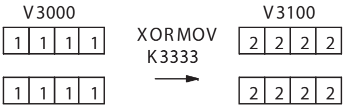
Load the constant value 2 (Hex) into the lower 16 bits of the accumulator.

Convert octal 3000 to HEX and load the value into the accumulator. This is the table beginning.

Load the constant value 8888 (Hex.) into the lower 16 bits of the accumulator.

Copy the table to V3100, ORing its contents with the accumulator as it is written.

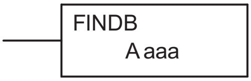
The example to the right shows a table of two words at V3000 and logically XORs it with K3333. The copy of the table at V3100 shows the result of the XOR operation for each word.



The ladder program example for the XORMOV is similar to the one above for the ORMOV. Just use the XORMOV instruction.

Find Block (FINDB)

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.



Operand Data Type		D4-454 Range
	A	aaa
V-memory	V	See memory map
V-memory	P	See memory map

Discrete Bit Flags	Description
SP53	On if there is a table pointer error.

The steps listed below are the steps necessary to program the Find Block function.

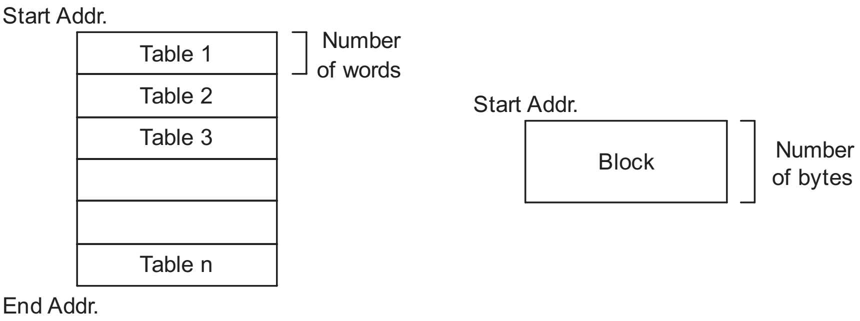
Step 1: Load the number of bytes in the block to be located. This parameter must be a HEX value, 0 to FF.

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 HEX value, 0 to FF.

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.

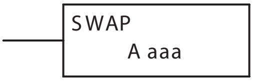
Step 4: Load the table starting location for all the tables into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

Step 5: Insert the Find Block instruction. This specifies the starting location of the block of data you are trying to locate.



Swap (SWAP)

The Swap instruction exchanges the data in two tables of equal length.



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.

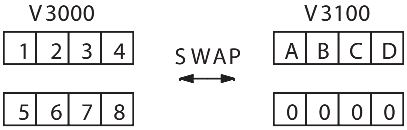
Step 2: Load the starting V-memory location for the first table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

Step 3: Insert the Swap instruction. This specifies the starting address of the second table. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

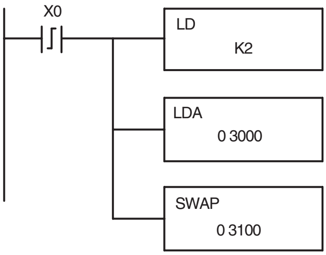
Helpful hint: The data swap occurs within a single scan. If the instruction executes on multiple consecutive scans, it will be difficult to know the actual contents of either table at any particular time. So, remember to swap just on a single scan.

Operand Data Type		D4-454 Range
		aaa
V-memory	V	See memory map

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 3100 by using the Swap instruction. The required ladder program is given below.



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.



Load the constant value 2 (Hex.) into the lower 16 bits of the accumulator.

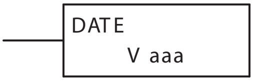
Convert octal 3000 to HEX and load the value into the accumulator. This is the table beginning.

Swap the contents of the table in the previous instruction with the one at V3100.

Clock/Calendar Instructions

Date (DATE)

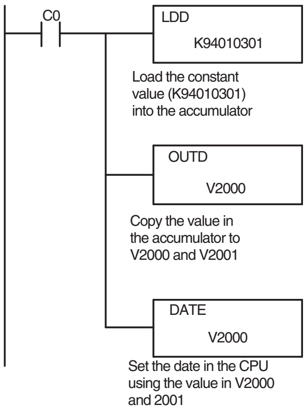
The Date instruction can be used to set the date in the CPU. The instruction requires two consecutive V-memory locations (Vaaa) to set the date. If the values in the specified locations are not valid, the date will not be set. The current date can be read from 4 consecutive V-memory locations (V7771 – V7774).

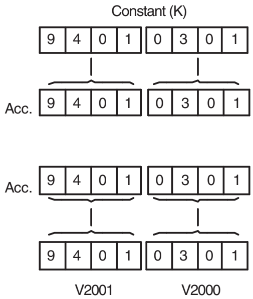


Date	Range	V-memory Location (BCD) (READ Only)
Year	0-99	V7774
Month	1-12	V7773
Day	1-31	V7772
Day of Week	0-06	V7771
The values entered for the day of week are: 0=Sunday, 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday		

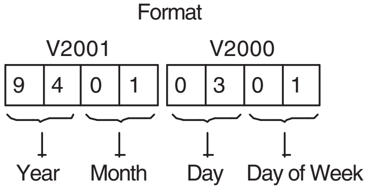
Operand Data Type		D4-454 Range
		aaa
V-memory	V	See memory map

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.


C0
LDD
K94010301
Load the constant value (K94010301) into the accumulator
OUTD
V2000
Copy the value in the accumulator to V2000 and V2001
DATE
V2000
Set the date in the CPU using the value in V2000 and 2001

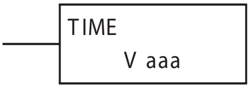


In this example, the Date instruction uses the value set in V2000 and V2001 to set the date in the appropriate V memory locations (V7771-V7774).



Time (TIME)

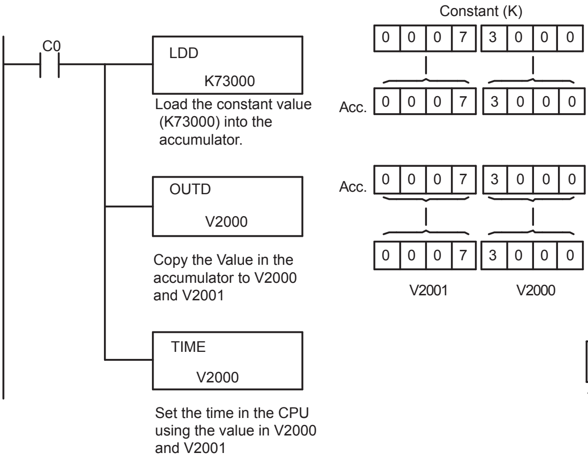
The Time instruction can be used to set the time (24 hour clock) in the CPU. The instruction requires two consecutive V-memory locations (Vaaa) which are used to set the time. If the values in the specified locations are not valid, the time will not be set. The current time can be read from memory locations V7747 and V7766 – V7770.



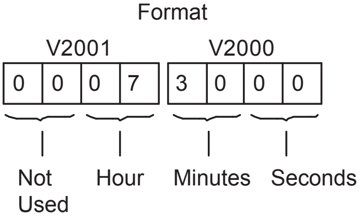
Date	Range	V-Memory Location (BCD) (READ Only)
1/100 seconds (10ms)	0-99	V7747
Seconds	0-59	V7766
Minutes	0-59	V7767
Hour	0-23	V7770

Operand Data Type	D4-454 Range
	aaa
V-memory V	See memory map

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.


C0
LDD
K73000
Load the constant value (K73000) into the accumulator.
OUTD
V2000
Copy the Value in the accumulator to V2000 and V2001
TIME
V2000
Set the time in the CPU using the value in V2000 and V2001
Constant (K)
0 0 0 7 3 0 0 0
Acc.
0 0 0 7 3 0 0 0
Acc.
0 0 0 7 3 0 0 0
0 0 0 7 3 0 0 0
V2001
V2000

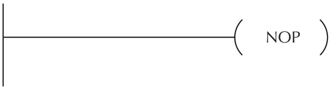
The TIME instruction uses the value set in V2000 and V2001 to set the time in the appropriate V-memory locations (V7766-V7770)



CPU Control Instructions

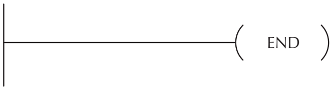
No Operation (NOP)

The No Operation is an empty (not programmed) memory location.



End (END)

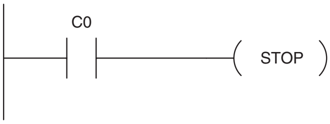
The End instruction marks the termination point of the normal program scan. An End instruction is required at the end of the main program body. If the End instruction is omitted, an error will occur and the CPU will not enter the Run Mode. Data labels, subroutines and interrupt routines are placed after the End instruction. The End instruction is not conditional; therefore, no input contact is allowed.



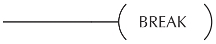
Stop (STOP)

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 an error condition.

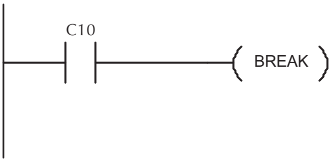
In the following example, when C0 turns on, the CPU will stop operation and switch to the program mode.



Break (BREAK)



The Break instruction changes the operational mode of the CPU from Run to Test Program mode. This instruction is typically used to aid in debugging an application program. The Break instruction allows V-memory and image register data to be retained where it would be normally cleared with the Stop instruction or a normal Run to Program transition.



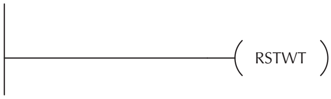
Reset Watch Dog Timer (RSTWT)

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 increased from the default value of 200ms with the DirectSOFT programming software. This eliminates the need for the RSTWT instruction.

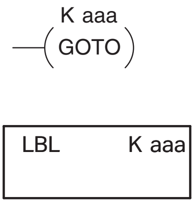
In the following example, the CPU scan timer will be reset to 0 when the RSTWT instruction is executed. See the For/Next instruction for a detailed example.



Program Control Instructions

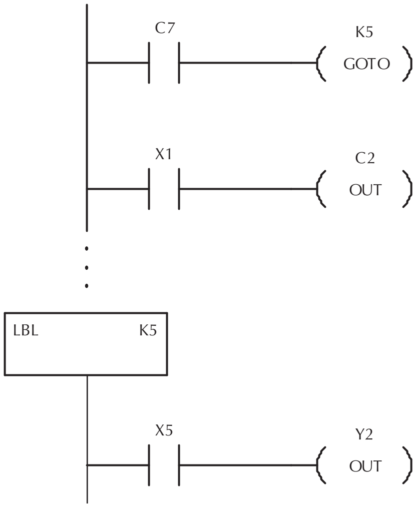
Goto Label (GOTO) (LBL)

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 are the same. The logic between Goto and LBL instruction is not executed when the Goto instruction is enabled. Up to 256 Goto instructions and 256 LBL instructions can be used in the program.

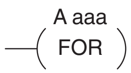


Operand Data Type		D4-454 Range
		aaa
Constant	K	1-256

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.

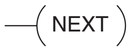


For / Next (FOR) (NEXT)



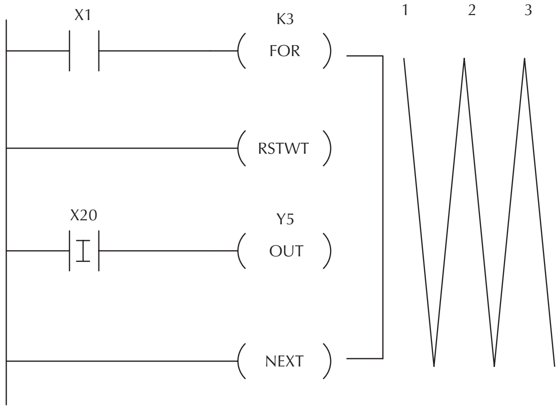
The For and Next instructions are used to execute a section of ladder logic between the For and Next instruction a specified numbers of times. When the For instruction is enabled, the program will loop the specified number of times. If the For instruction is not energized, the section of ladder logic between the For and Next instructions is not executed.

For / Next instructions cannot be nested. The normal I/O update and CPU housekeeping are suspended while executing the For / Next loop. The program scan can increase significantly, depending on the amount 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 watch dog timer inside of the For / Next loop using the RSTWT instruction.

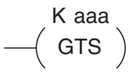


Operand Data Type		D4-454 Range
		aaa
V-memory	V	See memory map
Constant	K	1-9999

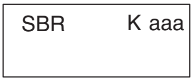
In the following example, when X1 is on, the application program inside the For / Next loop will be executed three times. If X1 is off, the program inside the loop will not be executed. The immediate instructions may or may not be necessary, depending on your application. Also, The RSTWT instruction is not necessary if the For / Next loop does not extend the scan time beyond the Watch Dog Timer setting. For more information on the Watch Dog Timer, refer to the RSTWT instruction.



Goto Subroutine (GTS) (SBR)



The Goto Subroutine instruction allows a section of ladder logic to be placed outside the main body of the program, to execute only when needed. You can have unlimited number of GTS instructions. Upon completion of executing the subroutine, program execution returns to the main program immediately after the GTS instruction. 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. There can be a maximum of 256 SBR instructions used 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 which 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	D4-454 Range
	aaa
Constant K	1-FFFF

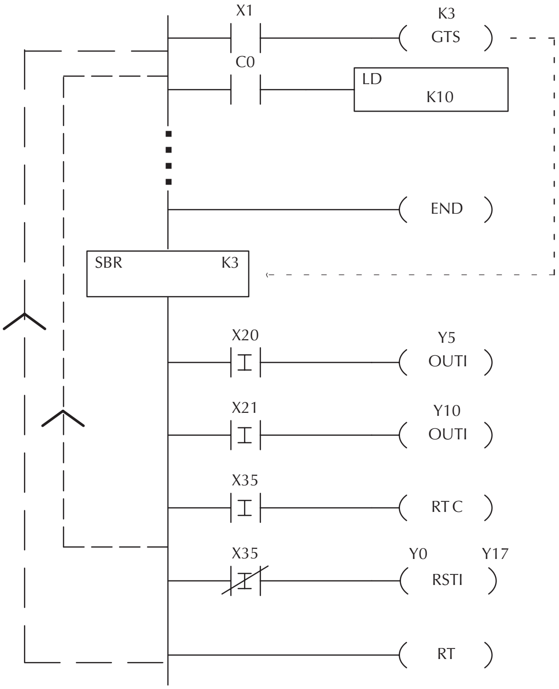
Subroutine Return (RT)

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. It must be the last instruction in the subroutine and is a stand alone instruction (no input contact on the rung).

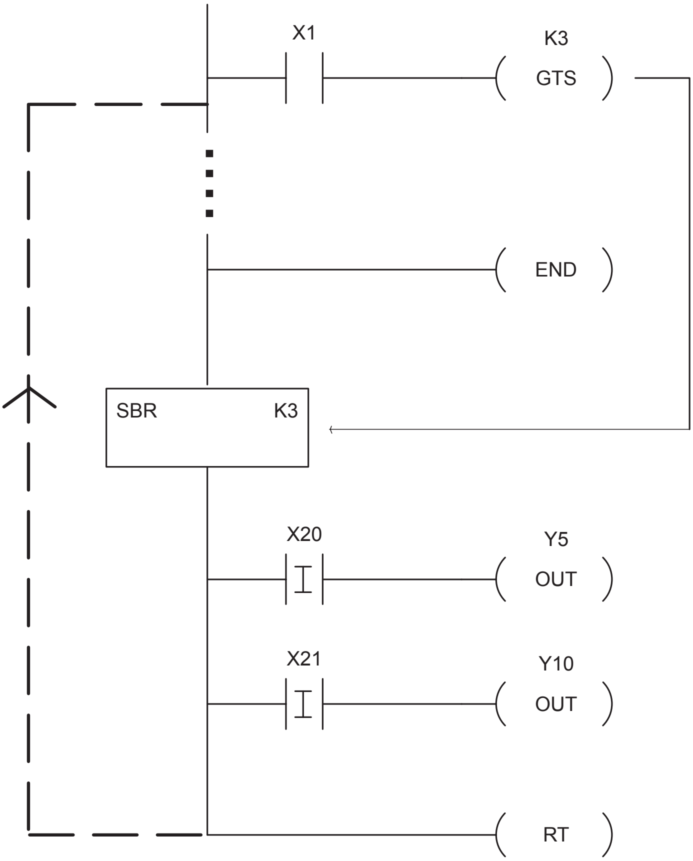
Subroutine Return Conditional (RTC)

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.

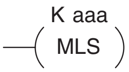
In the following example, when X1 is on, Subroutine K3 will be called. The CPU will jump to the Subroutine Label K3 and the ladder logic in the subroutine will be executed. If X35 is on, the CPU will return to the main program at the RTC instruction. If X35 is not on, Y0 – Y17 will be reset to off and the CPU will return to the main body of the program.



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.



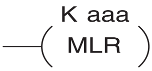
Master Line Set (MLS)



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.

Operand Data Type		D4-454 Range
		aaa
Constant	K	1-7

Master Line Reset (MLR)

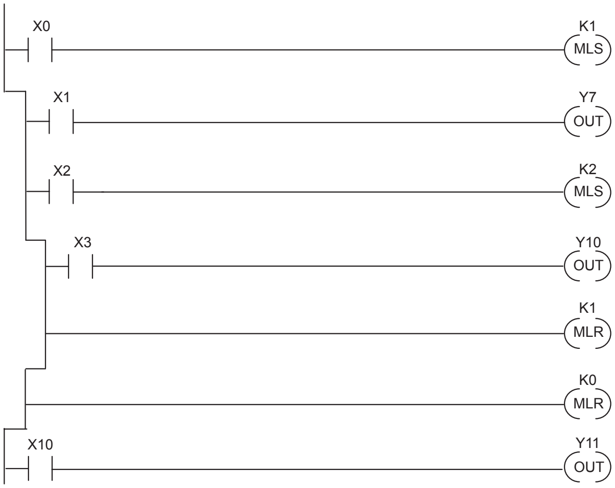


The Master Line Reset instruction marks the end of control for the corresponding MLS instruction. The MLR reference is one less than the corresponding MLS.

Operand Data Type		D4-454 Range
		aaa
Constant	K	0-6

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.



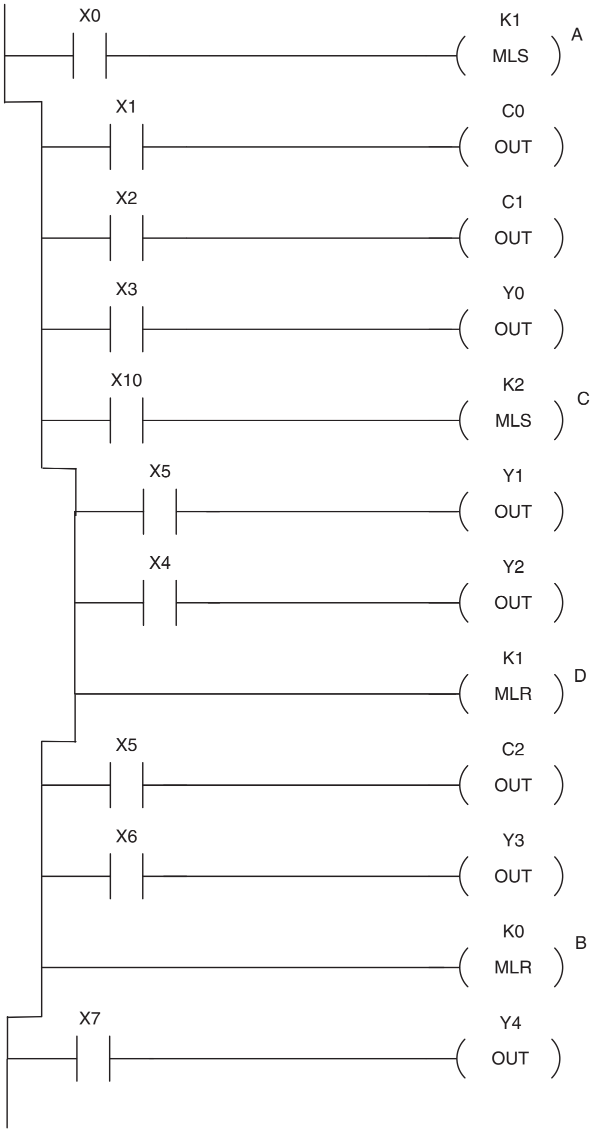
When contact X0 is ON, logic under the first MLS will be executed.

When contact X0 and X2 are ON, logic under the second MLS will be executed.

The MLR instructions note the end of the Master Control area.

MLS/MLR Example

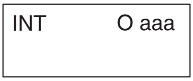
In the following MLS/MLR example logic between the first MLS K1 (A) and MLR K0 (B) will function only if input X0 is on. The logic between the MLS K2 (C) and MLR K1 (D) will function only if input X10 and X0 is on. The last rung is not controlled by either of the MLS coils.



Interrupt Instructions

Interrupt (INT)

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. 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, the CPU will complete execution of the instruction it is currently processing in ladder logic, then execute the interrupt routine.

There are two software interrupts and INT 16 and INT 17. This means that the hardware interrupts INT 16 and INT 17 and the software interrupts cannot be used at the same time.Once the interrupt is serviced, the program execution will continue from where it was before the interrupt occured. The software interrupts are setup by programming the interrupt times in V736 and V737. The valid range is 3 – 999ms. The value must be a BCD value. The interrupt will not execute if the value is out of range.

Operand Data Type		D4-454 Range
		aaa
Constant	0	0-17

Interrupt Return (IRT)

An Interrupt Return is normally executed as the last instruction in the interrupt routine. It returns the CPU to the point in the main program from which it was called. The Interrupt Return is a stand-alone instruction (no input contact on the rung).

Interrupt Return Conditional (IRTC)

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)

The Enable Interrupt instruction is placed in the main ladder program (before the End instruction), enabling the interrupt. The interrupt remains enabled until the program executes a Disable Interrupt instruction.

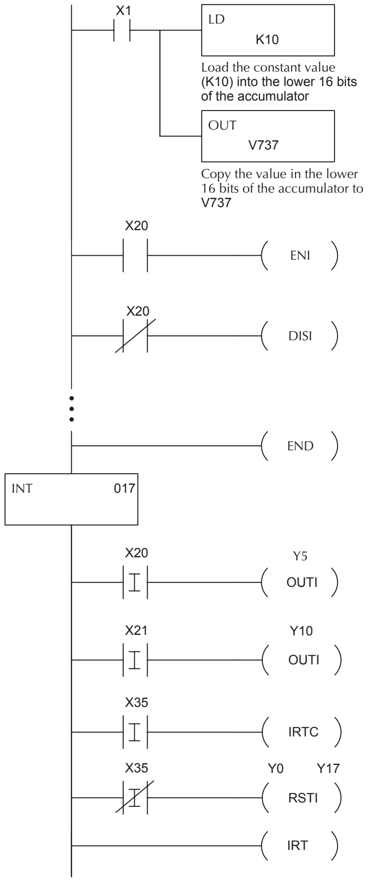
Disable Interrupts (DISI)

A Disable Interrupt instruction in the main body of the application program (before the End instruction) will disable the interrupt (either external or timed). The interrupt remains disabled until the program executes an Enable Interrupt instruction.

Interrupt Example for Software Interrupt

In the following example, when X1 is on, the value 10 is copied to V737. This value sets the software interrupt to 10 ms. When X20 turns on, the interrupt will be enabled. When X20 turns off, the interrupts will be disabled. Every 10 ms the CPU will jump to the interrupt label INT 017. The application ladder logic in the interrupt routine will be performed. If X35 is on the CPU will return to the main program with the IRTC instruction. If X35 is not on Y0 – Y17 will be reset to off and then the CPU will return to the program when the IRT instruction is executed. The software interrupt is limited to the range 3 – 999 milliseconds. Entering a 0, 1, or 2 will yield an interrupt time of 3 milliseconds.

***NOTE:** To take advantage of the speed of an Interrupt routine, use Immediate instructions for inputs and outputs. For example, Store immediate (STRI), Out immediate (OUTI) and Set immediate (SETI).*


X1
LD
K10
Load the constant value (K10) into the lower 16 bits of the accumulator
OUT
V737
Copy the value in the lower 16 bits of the accumulator to V737
X20
ENI
X20
DISI
END
INT
017
X20
Y5
OUTI
X21
Y10
OUTI
X35
IRTC
X35
Y0
Y17
RSTI
IRT

Message Instructions

System Errors and Fault Messages

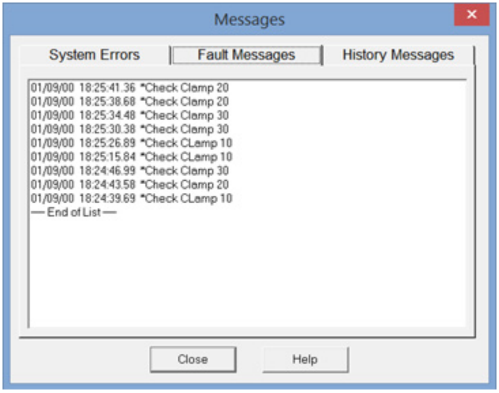
The D4-454 CPUs provide error logging capabilities. There are certain predefined system error messages and codes, but you can also use the Fault instruction to create your own specific messages. The CPU logs the error, the date, and the time the error occurred. There are two separate tables that store this information.

System Error Table -- The D4-454 can show up to 32 errors in an error table. When an error occurs, the error is loaded into the first available location. Therefore, the most recent error may not appear in the top row of the table. If the table is full when an error occurs, the oldest error is pushed (erased) from the table and the new error is inserted in the row.

Fault Message Table -- the D4-454 also allow you to build your own error codes and messages. These are called Fault Messages. You can build error codes, or up to 16 messages that can contain up to 23-character alphanumeric characters. In either case, you can have up to 16 messages or codes shown in the table. When a message is triggered, it is put in the first available table location. Therefore, the most recent error may not appear in the top row of the table. If the table is full when an error occurs, the oldest error is pushed (erased) from the table and the new error is inserted in the row.

The following diagram shows an example of a Fault Message table as shown in the DirectSOFT programming software.

There are several instructions that can be used in combination to create these error codes and fault messages.



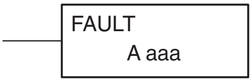
```
01/09/00  18:25:41.36 *Check Clamp 20
01/09/00  18:25:38.68 *Check Clamp 20
01/09/00  18:25:34.48 *Check Clamp 30
01/09/00  18:25:30.38 *Check Clamp 30
01/09/00  18:25:26.89 *Check CLamp 10
01/09/00  18:25:15.84 *Check CLamp 10
01/09/00  18:24:46.99 *Check Clamp 30
01/09/00  18:24:43.58 *Check Clamp 20
01/09/00  18:24:39.69 *Check CLamp 10
— End of List —
```

- FAULT -- Fault
- DLBL -- Data Label
- ACON -- ASCII Constant
- NCON -- Numeric Constant

The next few pages provide details on these instructions. Also, at the end of this section, there are two examples that show how the instructions are used together.

Fault (FAULT)

The Fault instruction is used to display a message or numeric error code in DirectSOFT, or on an Operator Interface Panel. The message has a maximum of 23 characters and can be either V-memory data, numerical constant data or ASCII text.



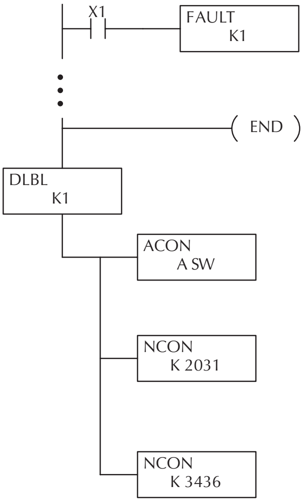
To display the value in a V-memory location, specify the V-memory location in the instruction. To display ASCII or numeric data, you must use the DLBL (Data Label) instruction, and ACON (ASCII constant) or NCON (Numeric constant) instructions, in conjunction with the Fault instruction. In this case, you should specify the constant (K) value in the Fault instruction for the corresponding data label area that contains the ACON and/or NCON instructions.

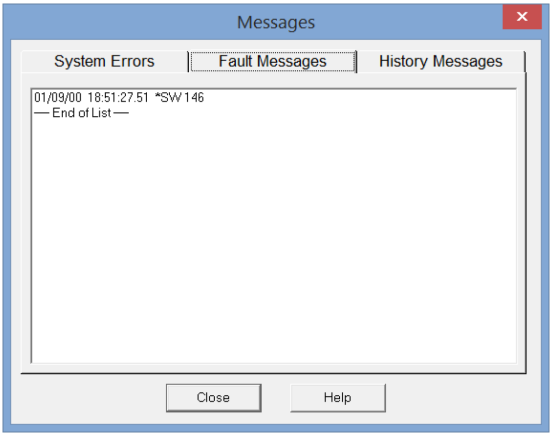
Operand Data Type		D4-454 Range
		aaa
V-memory	V	See memory map
Constant	K	1-FFFF

Discrete Bit Flags	Description
SP50	On when the FAULT instruction is executed

Fault Example

In the following example when X1 is on, the message SW 146 will be logged in the fault message table. The NCON use the HEX ASCII equivalent of the text to be displayed. (HEX ASCII for a blank is 20, a 1 is 31, 4 is 34 and 6 is 36).

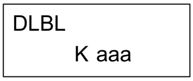




01/09/00 18:51:27.51 *SW 146
— End of List —

Data Label (DLBL)

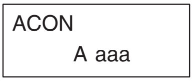
The Data Label instruction marks the beginning of an ASCII/numeric data area. DLBLs are programmed after the End statement. A maximum of 64 DLBL instructions can be used in a program. Multiple NCONs and ACONs can be used in a DLBL area.



Operand Data Type		D4-454 Range
		aaa
Constant	K	1-64

ASCII Constant (ACON)

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 a ACON a leading space will be inserted.

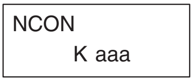


Operand Data Type		D4-454 Range
		aaa
ASCII	A	0-9 A-Z

When using DirectSOFT programming software, it is possible to store up to 40 characters in an ACON. See the ASCII Table Appendix for a complete listing of ASCII characters available. When the 40 characters are downloaded to the CPU, they are actually broken down into multiple ACONs that contain 2 characters each.

Numerical Constant (NCON)

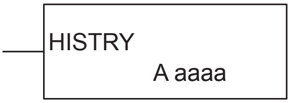
The Numerical Constant instruction is used with the DLBL instruction to store the HEX ASCII equivalent of numerical data for use with other instructions. Two digits can be stored in an NCON instruction.



Operand Data Type		D4-454 Range
		aaa
Constant	K	0-FFFF

History (HISTRY)

The History instruction stores event history information in memory of the PLC.

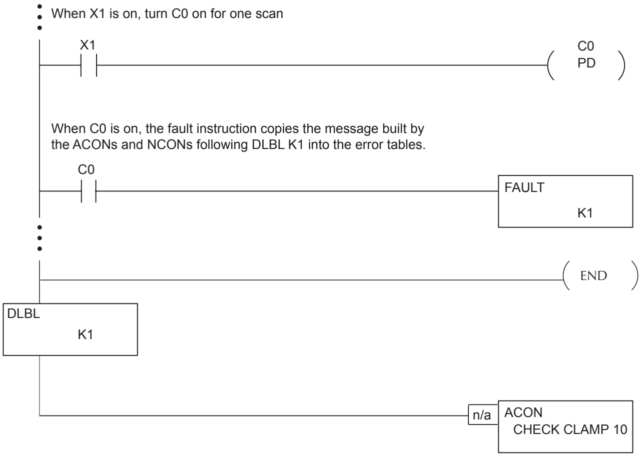


Important Information about FAULT Execution

It is important to understand how the Error Message and Error Code tables work when the Fault instruction is executed. Each time the instruction is executed, the code or message is inserted into the table. Since the CPU scan is extremely fast, then it is easily possible to completely fill up a table with a single message repeated each scan. In many cases this is not desirable, since it’s nice to maintain a history of the errors. (That’s why there are 32 positions in the Error Code table and 16 positions in the Error Message table.)

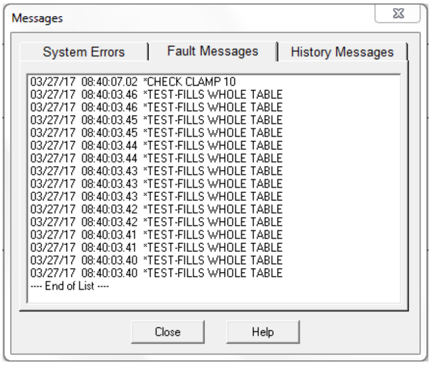
For example, let’s say you are examining a limit switch (X1). When the switch is closed, you want an error message (CHECK CLAMP 10) to be logged into the Error message table. In the real world, the limit switch may be closed for several seconds (or minutes, or hours...). During this time, the CPU will execute the Fault instruction a number of times, so the entire Error Message table will be full of a single message.

How do you solve this problem? Simple. Instead of using the limit switch to trigger the Fault instruction, use the limit switch to trigger a control relay used as a one-shot (PD coil instruction). Then, use the control relay as the input to the Fault instruction. Since the Fault is now triggered by the PD, which is only on for one scan, the message will only be copied into the table one time. This keeps the history of older messages intact.



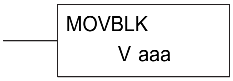
When X1 is on, turn C0 on for one scan

When C0 is on, the fault instruction copies the message built by the ACONs and NCONs following DLBL K1 into the error tables.



```
03/27/17  08:40:07.02  *CHECK CLAMP 10
03/27/17  08:40:03.46  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.46  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.45  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.45  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.44  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.44  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.43  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.43  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.43  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.42  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.42  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.41  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.41  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.40  *TEST-FILLS WHOLE TABLE
03/27/17  08:40:03.40  *TEST-FILLS WHOLE TABLE
---- End of List ----
```

Move Block (MOVBLK)



The Move Block instruction copies a specified number of words from a Data Label Area of program memory (ACON, NCON) to the specified V-memory location. Below are the steps for using the Move Block function:

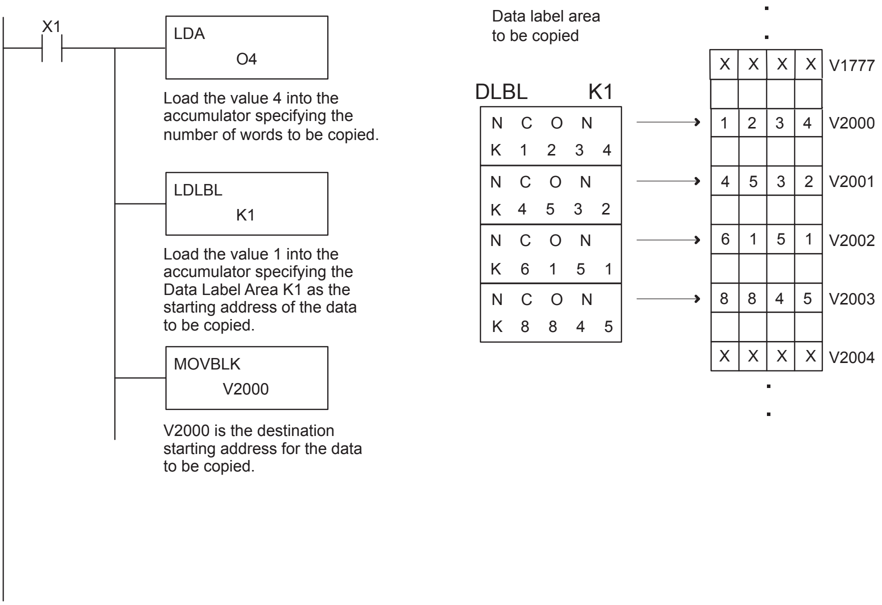
Step 1: Load the number of words (octal) to be copied into the 1st level of the accumulator stack.

Step 2: Load the source data label (LDLBL Kaaa) into the accumulator. This is where the data will be copied from.

Step 3: Insert the MOVBLK instruction that specifies the V-memory destination. This is where the data will be copied to.

Copy Data From a Data Label Area to V-memory

In the example below, data is copied from a Data Label Area to V-memory. When X1 is on, the octal value (O4) is copied to the first level of the accumulator stack using the Load Address (LDA) instruction. This value specifies the number of words to be copied. Load Label (LDLBL) instruction will load the source data address (K1) into the accumulator. This is where the data will be copied from. The MOVBLK instruction specifies the destination starting location and executes the copying of data from the Data Label Area to V-memory.



Load the value 4 into the accumulator specifying the number of words to be copied.

Load the value 1 into the accumulator specifying the Data Label Area K1 as the starting address of the data to be copied.

V2000 is the destination starting address for the data to be copied.

Print Message (PRINT)

The Print Message instruction prints the embedded text or text/data variable message (maximum 128 characters) to the specified communications port (Port 1, 2 and/or 3 on the D4-454 CPU), which must have the communications port configured.

PRINT A aaa
“Hello, this is a PLC message”

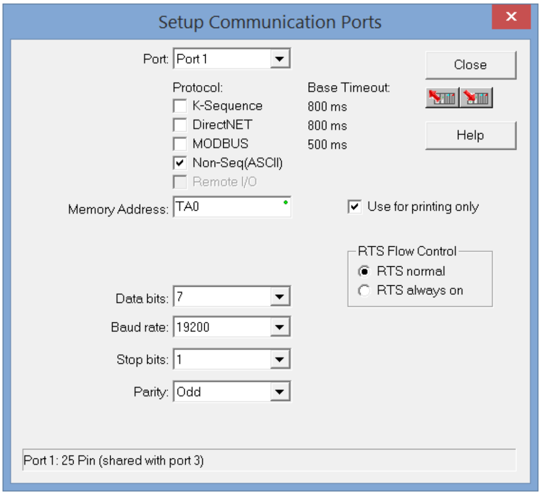
Operand Data Type		D4-454 Range
		aaa
Constant	K	1, 2 or 3

You may recall, from the CPU specifications in Chapter 3, that the DL454’s ports are capable of several protocols. Port 1, 2 and 3 can be configured for the non-sequence protocol. To configure a port in DirectSOFT, choose the PLC menu, then Setup, then Setup Secondary Comm Port.

For example,

Port: From the port number list box at the top, choose Port 2.

Protocol: Click the check box to the left of Non-sequence, and then you’ll see the dialog box shown below.



Setup Communication Ports

Port: Port 1

Protocol:
□ K-Sequence
□ DirectNET
□ MODBUS
☑ Non-Seq(ASCII)
□ Remote I/O

Base Timeout:
800 ms
800 ms
500 ms

Memory Address: TA0

☑ Use for printing only

RTS Flow Control
● RTS normal
○ RTS always on

Data bits: 7
Baud rate: 19200
Stop bits: 1
Parity: Odd

Port 1: 25 Pin (shared with port 3)

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.

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. IE V2000 to V2077. This area cannot be used by any other function in the ladder program.

Before ending the setup, click the button indicated to send the port configuration to the CPU, and click Close. See Chapter 3 for port wiring information, in order to connect your printer to the D4-454.

Text element – This part of the instruction is used to create the character string to be sent out of the port. 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.

Example:

”” Length 0 without character

”A” Length 1 with character A

” ” Length 1 with blank

” $” ” Length 1 with double quotation mark

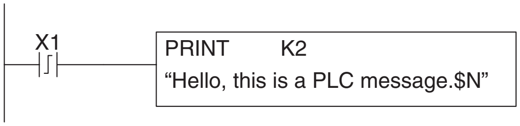
” $ R $ L ” Length 2 with one CR and one LF

” $ 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 E499 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 output device. 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 - this is 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 E499.*

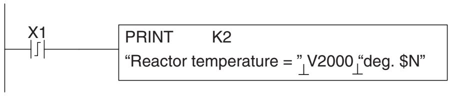
Example:

#	Character code	Description
1	None	16-bit binary (decimal number)
2	:B	4 digit BCD
3	:D	32-bit binary (decimal number)
4	:DB	8 digit BCD
5	:R	Floating point number (real number)
6	:E	Floating point number (real number with exponent)

V2000	Prints binary data in V2000 for decimal number
V2000:B	Prints BCD data in V2000
V2000:D	Prints binary number in V2000 and V2001 for decimal number
V2000:DB	Prints BCD data in V2000 and V2001
V2000:R	Prints floating point number in V2000/V2001 as real number
V2000:E	Prints 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.

V-memory text element - This is used for printing text stored in V-memory. 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 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.



Print the message to Port 2 when X1 makes an off-to-on transition.

⊥ represents a space

Message will read:
Reactor temperature = 0156 deg.

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

This is 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:ONOFF	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:

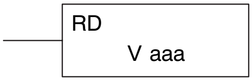
Special relay flags for port 1, 2, and 3 indicate the status of the D4-454 CPU ports (busy, or communications error). See the Special Relays appendix for a description.

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)	12
Floating point (real with exponent)	12
V-memory/text	2
Bit (1/0 format)	1
Bit (TRUE/FALSE format)	5
Bit (ON/OFF format)	3

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.*

Intelligent I/O Instructions

Read from Intelligent Module (RD)



The Read from Intelligent Module instruction reads a block of data (1-128 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.

Step 1: Load the base number (0-3) into the first byte and the slot number (0-7) 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 which specifies the starting V-memory location (Vaaa) where the data will be read into.

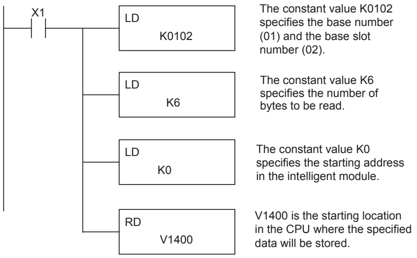
Helpful Hint: Use the LDA instruction to convert an octal address to its HEX equivalent and load it into the accumulator when the HEX format is required.

Operand Data Type		D4-454 Range
		aaa
V-memory	V	See memory map

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 a 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.

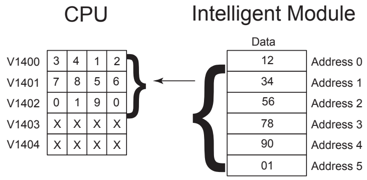


The constant value K0102 specifies the base number (01) and the base slot number (02).

The constant value K6 specifies the number of bytes to be read.

The constant value K0 specifies the starting address in the intelligent module.

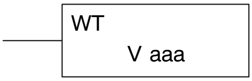
V1400 is the starting location in the CPU where the specified data will be stored.



CPU				
V1400	3	4	1	2
V1401	7	8	5	6
V1402	0	1	9	0
V1403	X	X	X	X
V1404	X	X	X	X

Data	
12	Address 0
34	Address 1
56	Address 2
78	Address 3
90	Address 4
01	Address 5

Write to Intelligent Module (WT)



The Write to Intelligent Module instruction writes a block of data (1-128 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 to program the Write to Intelligent module function.

Step 1: Load the base number (0-3) into the first byte and the slot number (0-7) 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.

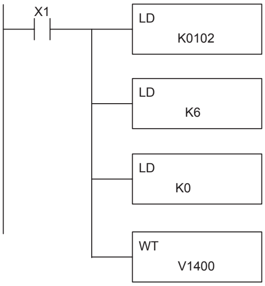
Helpful Hint: Use the LDA instruction to convert an octal address to its HEX equivalent and load it into the accumulator when the HEX format is required.

Operand Data Type	D4-454 Range
	aaa
V-memory V	See memory map

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 data from V-memory locations V1400-V1402.

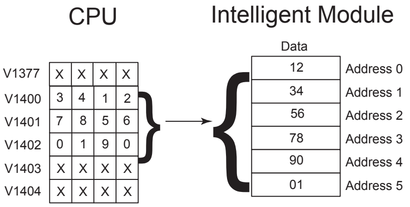


The constant value K0102 specifies the base number (01) and the base slot number (02).

The constant value K6 specifies the number of bytes to be written.

The constant value K0 specifies the starting address in the intelligent module.

V1400 is the starting location in the CPU where the specified data will be written from.



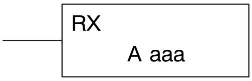
V1377	X	X	X	X
V1400	3	4	1	2
V1401	7	8	5	6
V1402	0	1	9	0
V1403	X	X	X	X
V1404	X	X	X	X

Data	
12	Address 0
34	Address 1
56	Address 2
78	Address 3
90	Address 4
01	Address 5

Network Instructions

Read from Network (RX)

The Read from Network instruction is used by the master device on a network to read a block of data from a slave device on the same network. 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 Network function.

Step 1: Load the slave address (0–90 BCD) into the first byte and the PLC internal port (Kf2) or slot number of the master DCM or ECOM (0–7) 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.

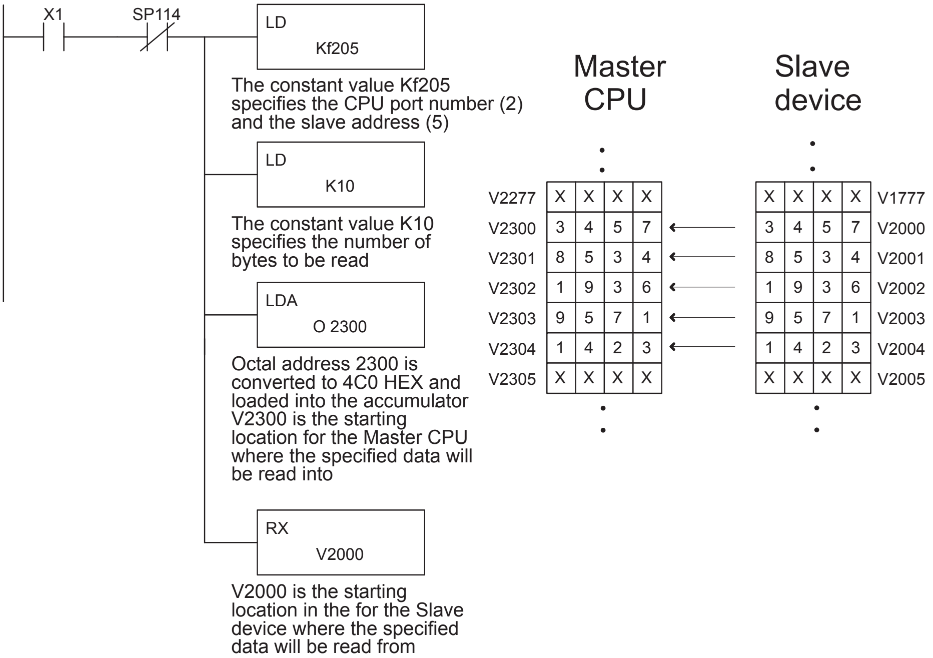
Step 3: Load the address of the data to be read into the accumulator. This parameter requires a HEX value.

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		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Inputs	X	0 – 1777
Outputs	Y	0 – 1777
Control Relays	C	0 – 3777
Stage	S	0 – 1777
Timer	T	0 – 377
Counter	CT	0 – 377
Special Relay	SP	0 – 777
Global I/O	GX	0 – 377
Program Memory	$	0 – 7680 (2K program mem.)

In the following example, when X1 is on and the port busy relay SP114 (see special relays) is not on, the RX instruction will access port 2 operating as a master. Ten consecutive bytes of data (V2000 – V2004) will be read from a slave device at station address 5 and copied into V-memory locations V2300 – V2304 in the CPU.



The constant value Kf205 specifies the CPU port number (2) and the slave address (5)

The constant value K10 specifies the number of bytes to be read

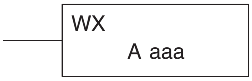
Octal address 2300 is converted to 4C0 HEX and loaded into the accumulator V2300 is the starting location for the Master CPU where the specified data will be read into

V2000 is the starting location in the for the Slave device where the specified data will be read from

Master CPU						Slave device				
V2277	X	X	X	X		X	X	X	X	V1777
V2300	3	4	5	7	←	3	4	5	7	V2000
V2301	8	5	3	4	←	8	5	3	4	V2001
V2302	1	9	3	6	←	1	9	3	6	V2002
V2303	9	5	7	1	←	9	5	7	1	V2003
V2304	1	4	2	3	←	1	4	2	3	V2004
V2305	X	X	X	X		X	X	X	X	V2005

Write to Network (WX)

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 accumulator and the first and second levels of the stack.



Listed below are the program steps necessary to execute the Write to Network function.

Step 1: Load the slave address (0–90 BCD) into the low byte and “Kf2” into the high byte of the accumulator (the next two instructions push this word down to the second layer of the stack).

Step 2: Load the number of bytes to be transferred into the accumulator (the next instruction pushes this word onto the top of the stack).

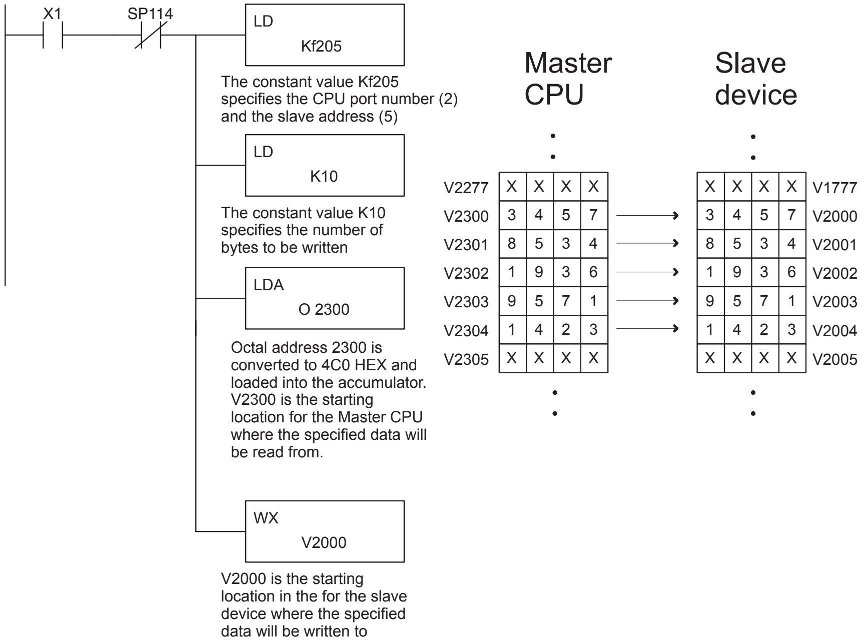
Step 3: Load the starting Master CPU address into the accumulator. This is the memory location where the data will be written from. This parameter requires a HEX value.

Step 4: Insert the WX instruction which specifies the starting V-memory location (Aaaa) where the data will be written to 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		D4-454 Range
	A	aaa
V-memory	V	See memory map
Pointer	P	See memory map
Inputs	X	0–1777
Outputs	Y	0–1777
Control Relays	C	0–3777
Stage	S	0–1777
Timer	T	0–377
Counter	CT	0–377
Special Relay	SP	0–777
Global I/O	GX	0–377
Program Memory	$	0–7680 (2K program mem.)

In the following example, when X1 is on and the module busy relay SP114(see special relays) is not on, the WX instruction will access port 2 operating as a master. Ten consecutive bytes of data are read from the Master CPU and copied to V-memory locations V2000–V2004 in the slave device at station address 5.



The constant value Kf205 specifies the CPU port number (2) and the slave address (5)

The constant value K10 specifies the number of bytes to be written

Octal address 2300 is converted to 4C0 HEX and loaded into the accumulator. V2300 is the starting location for the Master CPU where the specified data will be read from.

V2000 is the starting location in the for the slave device where the specified data will be written to

Master CPU				
V2277	X	X	X	X
V2300	3	4	5	7
V2301	8	5	3	4
V2302	1	9	3	6
V2303	9	5	7	1
V2304	1	4	2	3
V2305	X	X	X	X

Slave device				
X	X	X	X	V1777
3	4	5	7	V2000
8	5	3	4	V2001
1	9	3	6	V2002
9	5	7	1	V2003
1	4	2	3	V2004
X	X	X	X	V2005

DRUM INSTRUCTION PROGRAMMING

CHAPTER 6

In This Chapter...

Introduction

Purpose

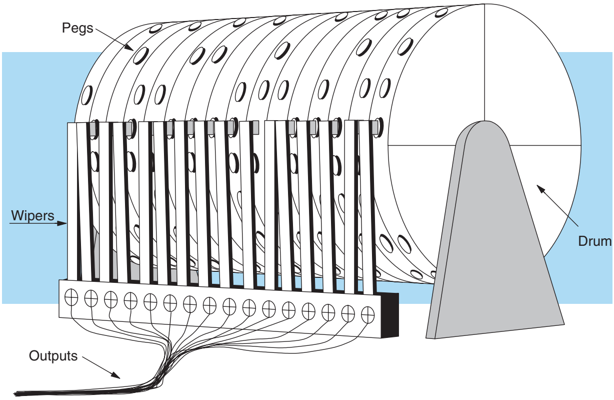
The four drum instructions in the D4-454 CPU electronically stimulate an electro-mechanical drum sequencer. The instruction offers enhancements to the basic principle, which we describe first.

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 power up, 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!

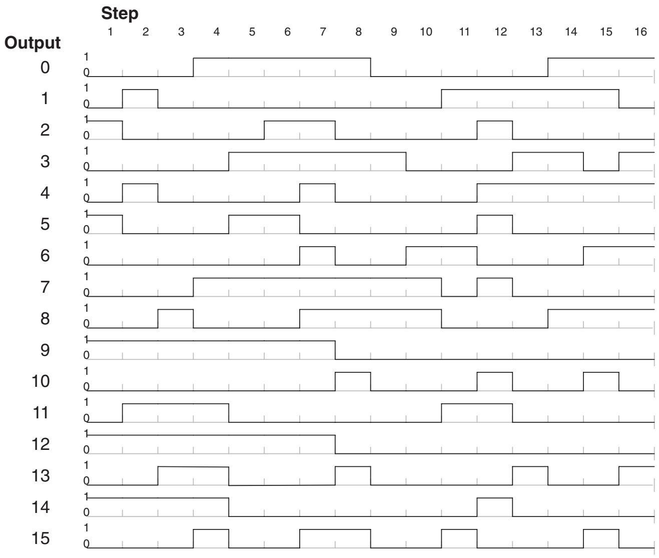
Drum Chart Representation

For editing purposes, the electronic drum is presented in chart form in DirectSOFT 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 D4-454 CPU:

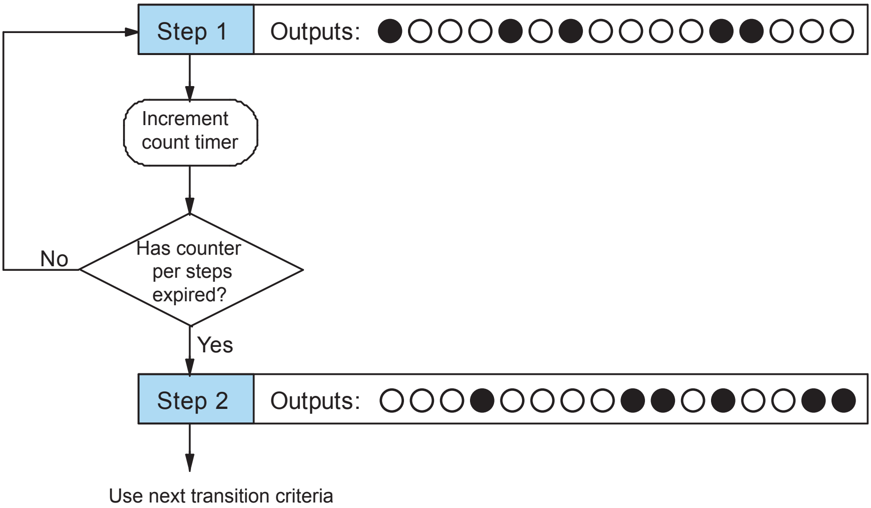
- Timed Drum with Discrete Outputs (DRUM)
- Time and Event Drum with Discrete Outputs (EDRUM)
- Masked Even Drum with Discrete Outputs (MDRMD)
- Masked Event Drum with Word Output (MDRMW)

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).

Timer-Only Transitions

Drums move from one step to another based on time. 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 stays 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. 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.


Step 1
Outputs: ●○○○●○●○○○○●●○○○
Increment count timer
Has counter per steps expired?
No
Yes
Step 2
Outputs: ○○○●○○○○●●○●○○●●
Use next transition criteria

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:

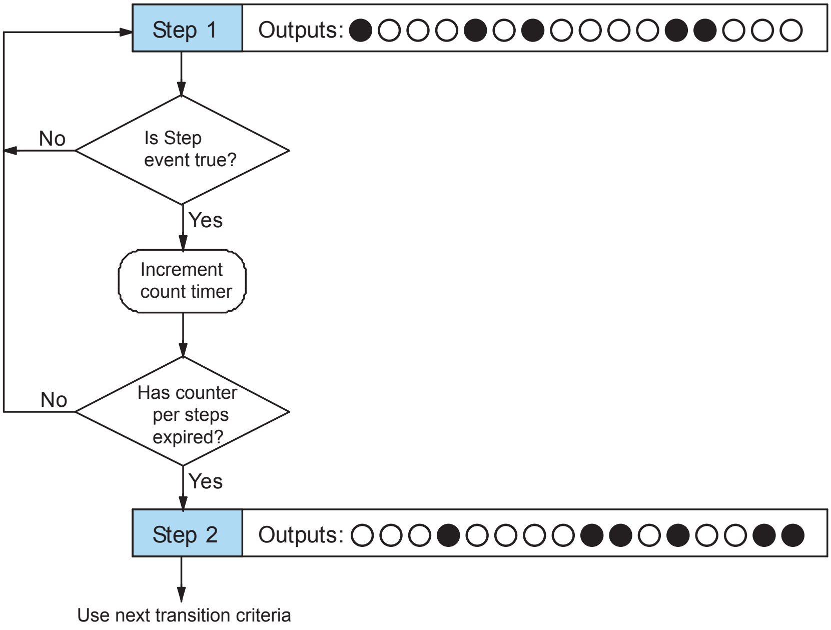
Max Time per step = 0.01 seconds X 9999 X 9999

= 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

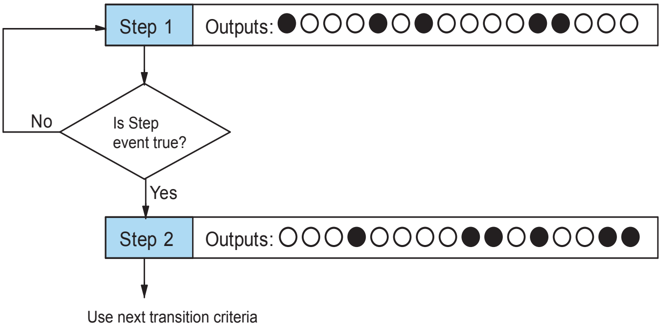
Step transitions may also occur based on time and/or external events. The figure below shows how step transitions work in these cases.


Step 1
Outputs: ●○○○●○●○○○○●●○○○
Is Step event true?
No
Yes
Increment count timer
Has counter per steps expired?
No
Yes
Step 2
Outputs: ○○○●○○○○●●○●○○●●
Use next transition criteria

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.


Step 1
Outputs: ●○○○●○●○○○○●●○○○
No
Is Step event true?
Yes
Step 2
Outputs: ○○○●○○○○●●○●○○●●
Use next transition criteria

Counter Assignments

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.

NOTE: *Each drum instruction uses the resources of four counters in the CPU..*

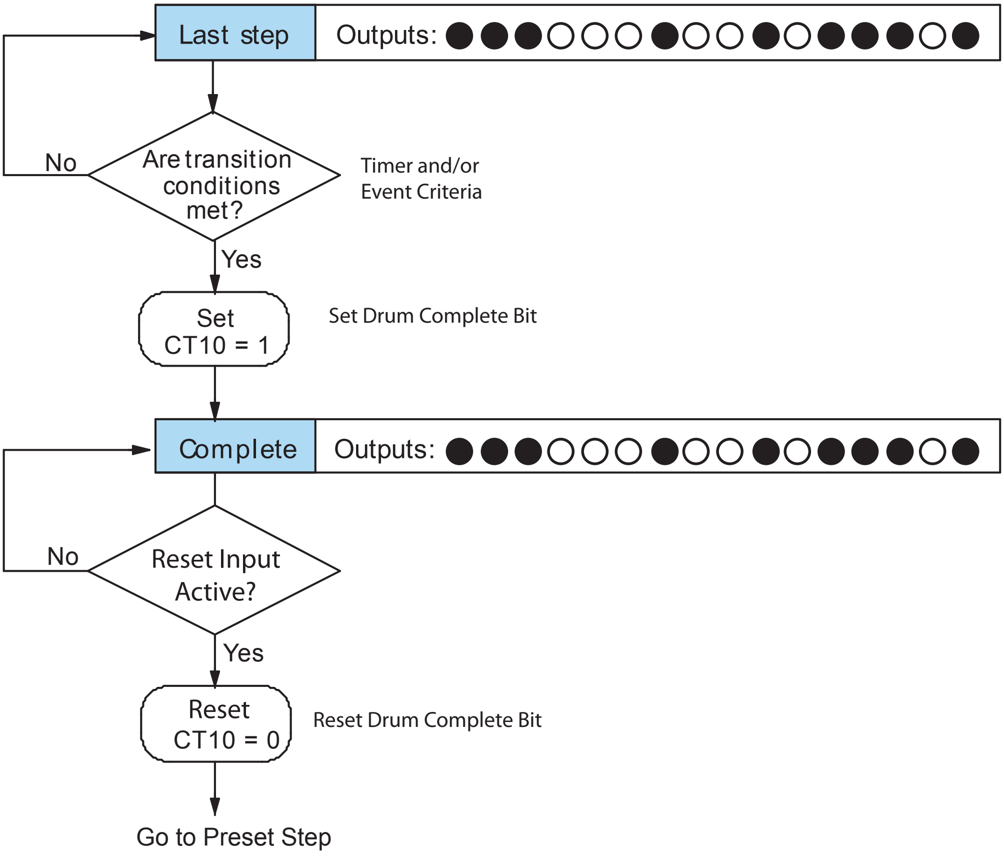
Suppose we program a timer drum to have 8 steps, and we select CT10 for the counter number (remember, counter numbering is in octal). Counter usage is shown to the right. The right column holds typical values, interpreted below.

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

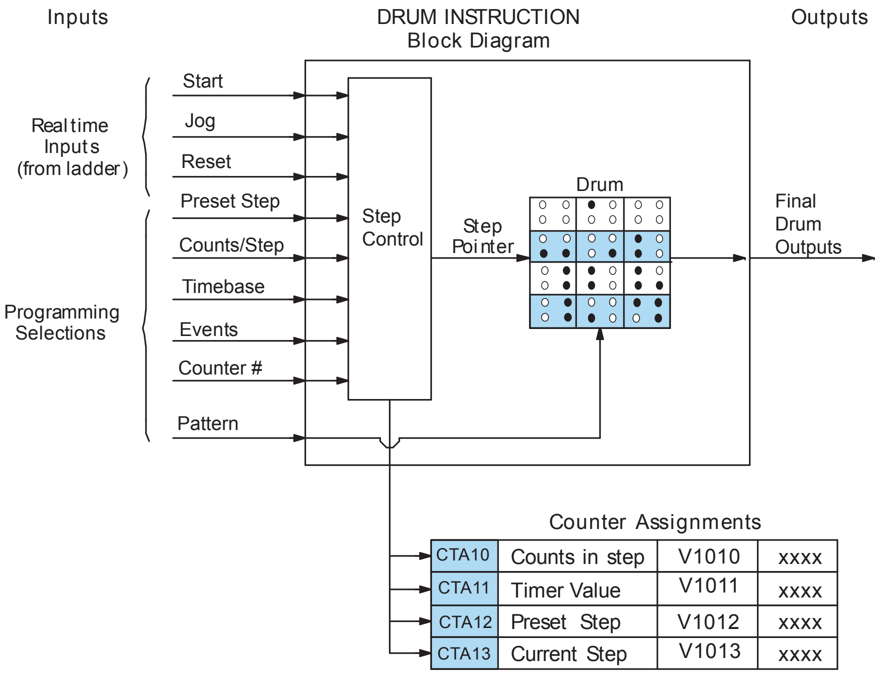
The last step in a drum sequence may be any step number, since partial drums are valid. Refer to the following figure. When the transition conditions of the last step are met, the drum sets the counter bit corresponding to the counter named in the drum instruction box (such as CT10). Then it moves to 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.


Last step
Outputs: ● ● ● ○ ○ ○ ● ○ ○ ● ○ ● ● ● ○ ●
No
Are transition conditions met?
Timer and/or Event Criteria
Yes
Set CT10 = 1
Set Drum Complete Bit
Complete
Outputs: ● ● ● ○ ○ ○ ● ○ ○ ● ○ ● ● ● ○ ●
No
Reset Input Active?
Yes
Reset CT10 = 0
Reset Drum Complete Bit
Go to 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.



CTA10	Counts in step	V1010	xxxx
CTA11	Timer Value	V1011	xxxx
CTA12	Preset Step	V1012	xxxx
CTA13	Current Step	V1013	xxxx

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. The D4-454 has 256 counters (CT0 – CT377 in octal).
- Events – Either an X, Y, C, GX, GY, S, CT, or SP type discrete point serves as step transition inputs. Each step has its own event. However, programming the event is optional.

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.

Power Up State of Drum Registers

The choice of the starting step on power up 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 power up or program-to-run mode transition. However, if the counter memory is configured to be retentive, the drum will stay in its previous state.

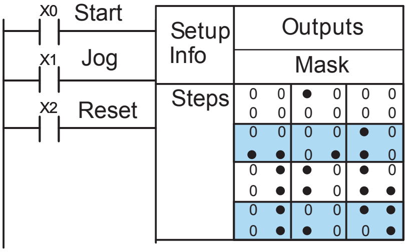
Counter Number	Function	Initialization on Power UP	
		Non-Retentive Case	Retentive Case
CTA(n)	Current Step Count	Initialize = 0	Use Previous (no change)
CTA(n + 1)	Counter Timer Value	Initialize = 0	Use Previous (no change)
CTA(n + 2)	Preset Step	Initialize = Preset Step #	Use Previous (no change)
CTA(n + 3)	Current Step #	Initialize = Preset Step #	Use Previous (no change)

Applications with relatively fast drum cycle times typically will need to be reset on power up, 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

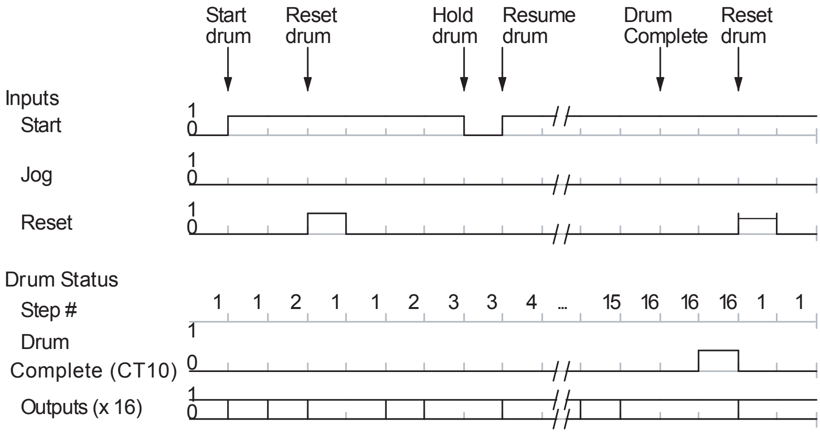
Now we are ready to put together the concepts on the previous pages and demonstrate general control of the drum instruction box. The drawing to the right shows a simplified generic drum instruction. Inputs from ladder logic control the Start, Jog, and Reset Inputs (only the EDRUM instruction supports 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.

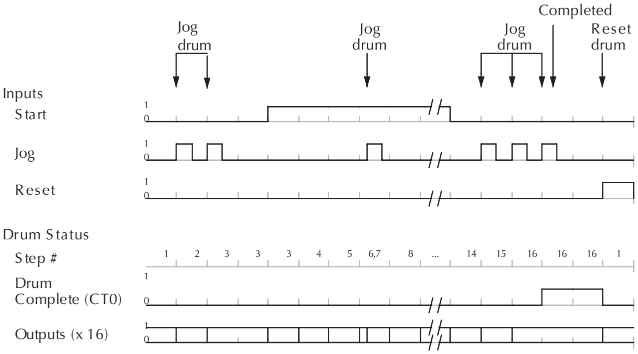


When the drum completes the last step (Step 16 in this example), the Drum Complete bit (CT10) turns on, and the step number remains at 16. When the Reset input turns on, it turns off the Drum Complete bit (CT10), and forces the drum to enter the preset step.

NOTE: *The timing diagram shows all steps using equal time durations. Step times can vary greatly, depending on the counts/step programmed.*

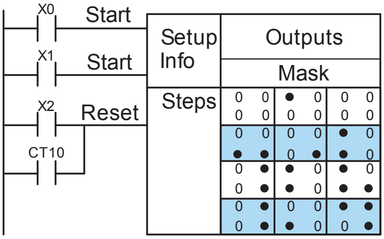
In the figure below, we focus on how the Jog input works on event drums. To the left of the diagram, note that the off-to-on transitions of the Jog input increments the step. Start may be either on or off (however, Reset must be off). Two jogs takes the drum to step three. Next, the Start input turns 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 power up, make the preset step in the drum a "reset step", with all outputs off.

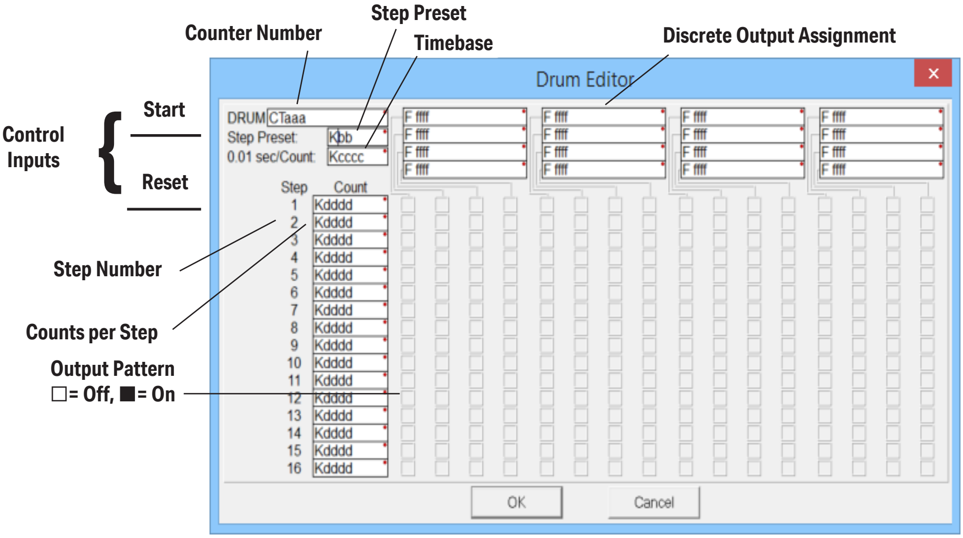
Using Complex Event Step Transitions

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

Timed Drum with Discrete Outputs (DRUM)

The Timed Drum with Discrete Outputs is the most basic of the D4-454's drum instructions. It operates according to the principles covered on the previous pages. Below is the instruction in chart form as displayed by DirectSOFT.



DRUM CTaaa
Step Preset: Kbb
0.01 sec/Count: Kcccc

Step	Count
1	Kdddd
2	Kdddd
3	Kdddd
4	Kdddd
5	Kdddd
6	Kdddd
7	Kdddd
8	Kdddd
9	Kdddd
10	Kdddd
11	Kdddd
12	Kdddd
13	Kdddd
14	Kdddd
15	Kdddd
16	Kdddd

F ffff (×16)

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, C, GX or GY types, or may be left unused. The output pattern may be edited graphically with DirectSOFT.

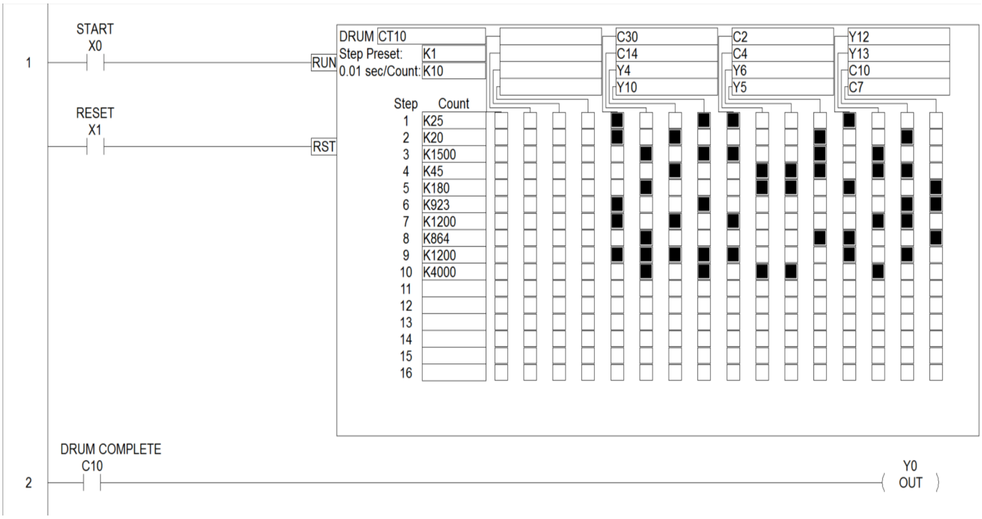
Whenever the Start input is energized, the drum's timer is enabled. It stops when the last step is complete, or when the Reset input is energized. The drum enters the preset step chosen upon a CPU program-to-run mode transition, and whenever the Reset input is energized.

Drum Parameters	Field	Data Types	Ranges
Counter Number	CTaaa	0 – 174	0 – 377
Preset Step	Kbb	K	1 – 16
Timer base	Kcccc	K	0 – 99.99 seconds
Counts per step	Kdddd	K	0 – 9999
Discrete Outputs	Fffff	X, Y, C, GX, GY	See memory map

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	Ranges of (n)	Function	Counter Bit Function
CTA(n)	0 – 374	Counts in step	CT(n) = Drum Complete
CTA(n+1)	1 – 375	Timer value	CT(n+1) = (not used)
CTA(n+2)	2 – 376	Preset Step	CT(n+2) = (not used)
CTA(n+3)	3 – 377	Current Step	CT(n+3) = (not used)

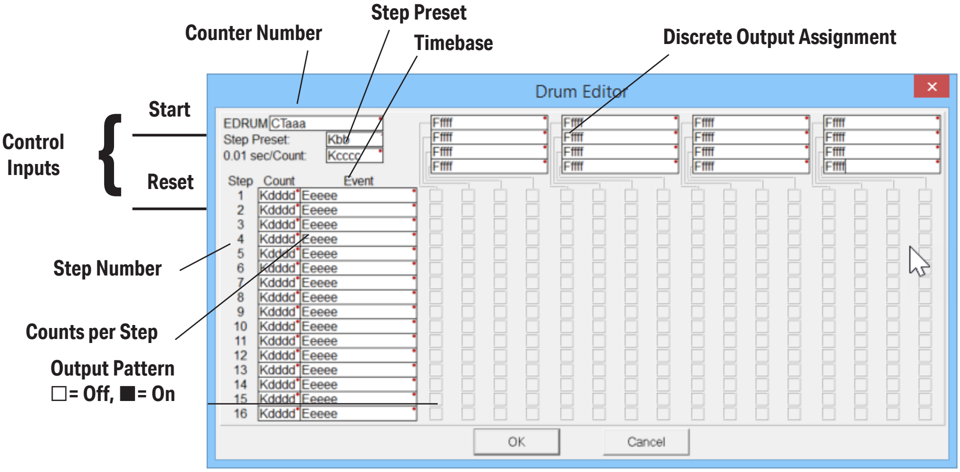
The following ladder program shows the DRUM instruction in a typical ladder program, as shown by DirectSOFT. Steps 1 through 10 are used, and twelve of the sixteen output points are used. The preset step is step 1. The timebase runs at (K10 x 0.01) = 0.1 second per count. Therefore, the duration of step 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.



Step	Count					C30	C14	Y4	Y10	C2	C4	Y6	Y5	Y12	Y13	C10	C7
1	K25	□	□	□	□	■	□	□	■	■	□	□	□	■	□	□	□
2	K20	□	□	□	□	■	□	■	□	□	□	□	■	□	□	■	□
3	K1500	□	□	□	□	□	■	□	■	■	□	□	■	□	■	□	□
4	K45	□	□	□	□	□	□	■	□	□	■	■	■	□	■	■	□
5	K180	□	□	□	□	□	■	□	□	□	■	■	□	■	□	□	■
6	K923	□	□	□	□	■	□	□	■	□	□	□	□	□	□	■	■
7	K1200	□	□	□	□	■	□	■	□	■	□	□	□	□	■	■	□
8	K864	□	□	□	□	□	■	□	□	□	□	□	■	■	□	□	■
9	K1200	□	□	□	□	■	■	■	■	■	□	□	□	■	□	■	□
10	K4000	□	□	□	□	□	■	□	■	□	■	■	□	□	■	□	□
11		□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
12		□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
13		□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
14		□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
15		□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
16		□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□

Event Drum (EDRUM)

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 DirectSOFT programming software.



EDRUM CTaaa
Step Preset: Kbb
0.01 sec/Count: Kcccc

Step	Count	Event
1	Kdddd	Eeeee
2	Kdddd	Eeeee
3	Kdddd	Eeeee
4	Kdddd	Eeeee
5	Kdddd	Eeeee
6	Kdddd	Eeeee
7	Kdddd	Eeeee
8	Kdddd	Eeeee
9	Kdddd	Eeeee
10	Kdddd	Eeeee
11	Kdddd	Eeeee
12	Kdddd	Eeeee
13	Kdddd	Eeeee
14	Kdddd	Eeeee
15	Kdddd	Eeeee
16	Kdddd	Eeeee

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.

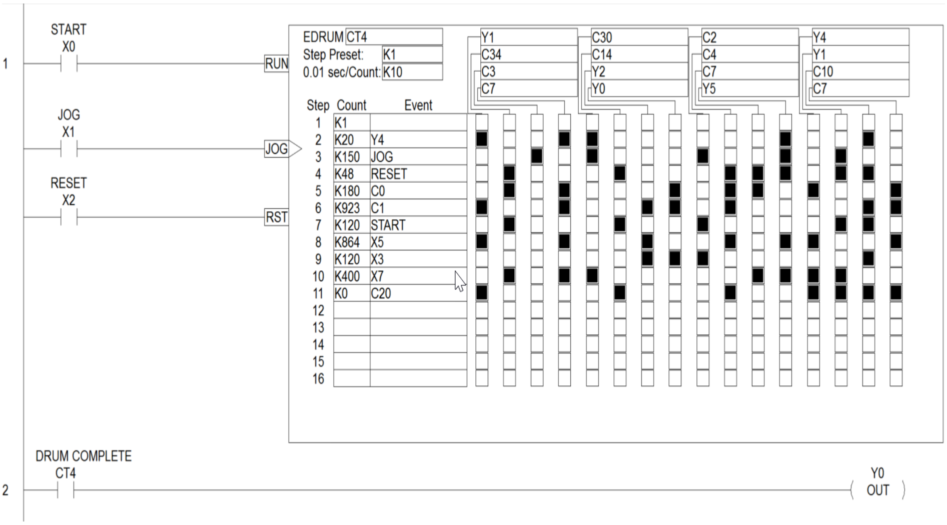
Whenever the Start input is energized, the drum’s timer is enabled. As long as the event is true for the current step, the timer runs during that step. When the step count equals the counts per step, the drum transitions to the next step. This process stops when the last step is complete, or when the 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	CTaaa	–	0 – 377
Preset Step	Kbb	K	1 – 16
Timer base	Kcccc	K	0 — 99.99 seconds
Counts per step	Kdddd	K	0 – 9999
Event	Eeeee	X, Y, C, GX, GY, S, T, CT, SP	See memory map
Discrete Outputs	Fffff	X, Y, C, GX, GY	See memory map

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	Ranges of (n)	Function	Counter Bit Function
CTA(n)	0 – 374	Counts in step	CT(n)= Drum Complete
CTA(n+1)	1 – 375	Timer value	CT(n+1) = (not used)
CTA(n+2)	2 – 376	Preset Step	CT(n+2) = (not used)
CTA(n+3)	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 DirectSOFT. Steps 1 through 11 are used, and all sixteen 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 power up 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.



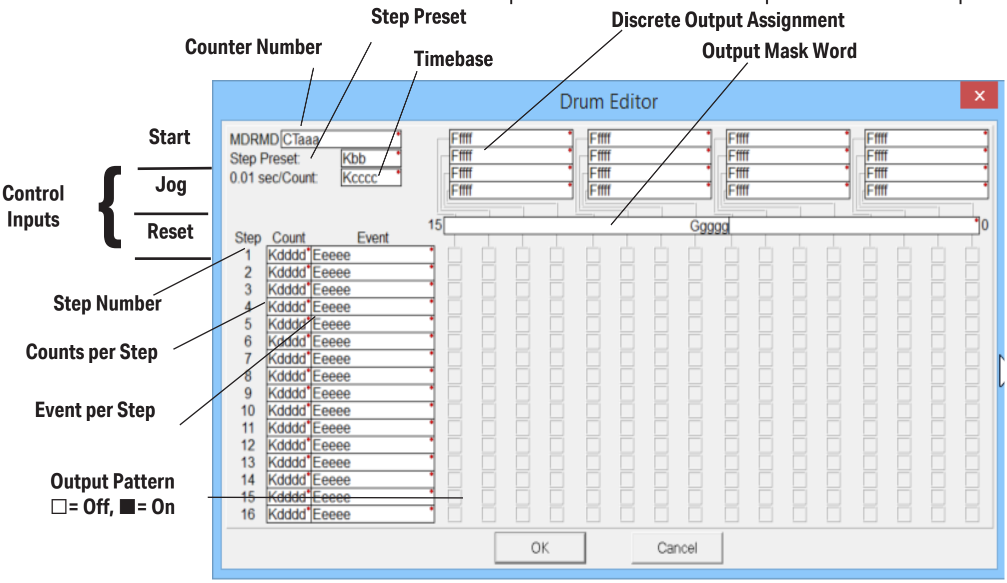
EDRUM CT4
Step Preset: K1
0.01 sec/Count: K10

Step	Count	Event	Y1	C34	C3	C7	C30	C14	Y2	Y0	C2	C4	C7	Y5	Y4	Y1	C10	C7
1	K1		□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
2	K20	Y4	■	□	□	■	■	□	□	□	□	□	□	■	□	□	■	□
3	K150	JOG	□	□	■	□	■	□	□	□	■	□	□	■	□	■	□	□
4	K48	RESET	□	■	□	□	□	■	□	□	□	■	■	■	□	■	■	□
5	K180	C0	□	■	□	■	□	□	□	■	□	■	■	□	■	□	□	■
6	K923	C1	■	□	□	■	□	□	■	■	□	■	□	□	□	□	■	■
7	K120	START	□	■	□	□	□	■	□	□	■	□	□	□	□	■	■	□
8	K864	X5	■	□	□	■	□	□	■	□	□	■	□	■	■	□	□	■
9	K120	X3	□	□	□	□	□	□	■	■	■	□	□	□	□	□	■	□
10	K400	X7	□	■	□	■	■	□	□	□	□	□	■	■	■	■	□	□
11	K0	C20	■	□	□	□	□	■	□	□	□	■	□	□	■	■	■	■
12			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
13			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
14			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
15			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
16			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□

Masked Event Drum with Discrete Outputs (MDRMD)

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 DirectSOFT programming software.

The Masked Event Drum with Discrete Outputs features sixteen steps and sixteen outputs.



MDRMD CTaaa
Step Preset: Kbb
0.01 sec/Count: Kcccc

Ffff	Ffff	Ffff	Ffff
Ffff	Ffff	Ffff	Ffff
Ffff	Ffff	Ffff	Ffff
Ffff	Ffff	Ffff	Ffff

Ggggg

Step	Count	Event
1	Kdddd	Eeeee
2	Kdddd	Eeeee
3	Kdddd	Eeeee
4	Kdddd	Eeeee
5	Kdddd	Eeeee
6	Kdddd	Eeeee
7	Kdddd	Eeeee
8	Kdddd	Eeeee
9	Kdddd	Eeeee
10	Kdddd	Eeeee
11	Kdddd	Eeeee
12	Kdddd	Eeeee
13	Kdddd	Eeeee
14	Kdddd	Eeeee
15	Kdddd	Eeeee
16	Kdddd	Eeeee

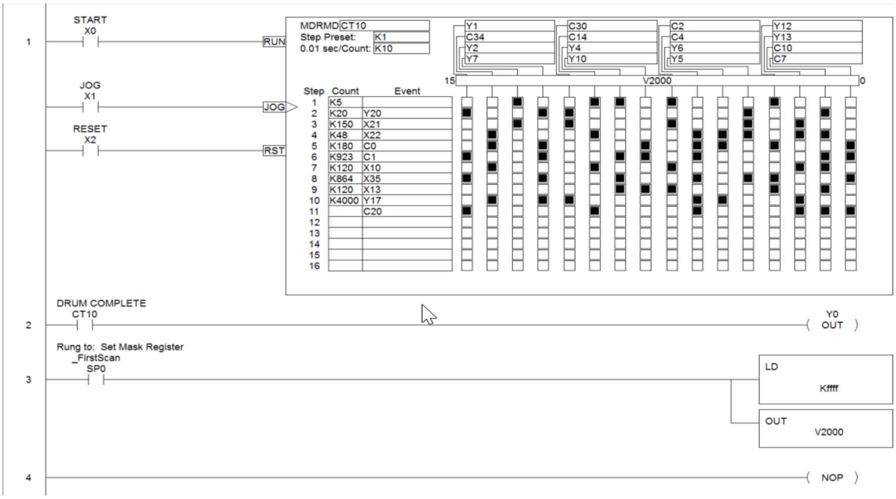
Drum outputs are logically ANDed bit-by-bit with an output mask word for each step. The Ggggg field specifies the beginning location of the 16 mask words. Step transitions occur on timed and/or event basis. The jog input also advances the step on each off-to-on transition. Time is specified in counts 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	CTaaa	–	0–377
Preset Step	Kbb	K	1–16
Timer base	Kcccc	K	0–99.99 seconds
Counts per step	Kdddd	K	0–9999
Event	Eeeee	X, Y, C, GX, GY. S, T, CT, SP	See memory map
Discrete Outputs	Fffff	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	Ranges of (n)	Function	Counter Bit Function
CTA(n)	0 – 374	Counts in step	CT(n) = Drum Complete
CTA(n+1)	1 – 375	Timer value	CT(n+1) = (not used)
CTA(n+2)	2 –376	Preset Step	CT(n+2) = (not used)
CTA(n+3)	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 DirectSOFT. 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 power up, 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 10). A drum reset also resets CT10.

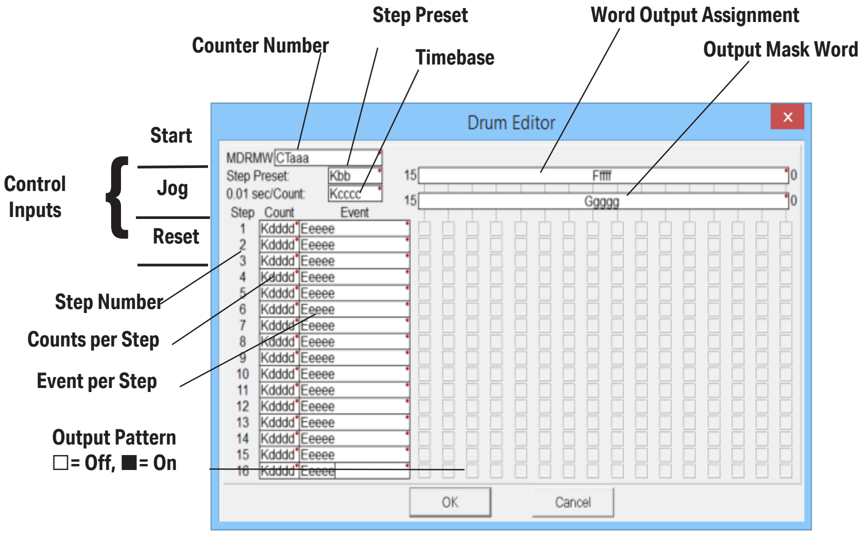


Step	Count	Event	15															0
1	K5		○	○	●	○	○	●	●	○	●	○	○	○	●	○	○	○
2	K20	Y20	●	○	○	●	●	○	○	○	○	○	○	●	○	○	●	○
3	K150	X21	○	○	●	○	●	○	○	○	●	○	○	●	○	●	○	○
4	K48	X22	○	●	○	○	○	●	○	○	○	●	●	●	○	●	●	○
5	K180	C0	○	●	○	●	○	○	○	●	○	●	●	○	●	○	○	●
6	K923	C1	●	○	○	●	○	○	●	●	○	●	○	○	○	○	●	●
7	K120	X10	○	●	○	○	○	●	○	○	●	○	○	○	○	●	●	○
8	K864	X35	●	○	○	●	○	○	●	○	○	●	○	●	●	○	○	●
9	K120	X13	○	○	○	○	○	○	●	●	●	○	○	○	●	○	●	○
10	K4000	Y17	○	●	○	●	●	○	○	○	○	●	●	○	○	●	○	○
11		C20	●	○	○	○	○	●	○	○	○	●	○	○	○	●	●	●
12			○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
13			○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
14			○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
15			○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
16			○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

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)

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 DirectSOFT.



MDRMW CTaaa
Step Preset: Kbb
0.01 sec/Count: Kcccc
15 Fffff 0
15 Ggggg 0

Step	Count	Event
1	Kdddd	Eeeee
2	Kdddd	Eeeee
3	Kdddd	Eeeee
4	Kdddd	Eeeee
5	Kdddd	Eeeee
6	Kdddd	Eeeee
7	Kdddd	Eeeee
8	Kdddd	Eeeee
9	Kdddd	Eeeee
10	Kdddd	Eeeee
11	Kdddd	Eeeee
12	Kdddd	Eeeee
13	Kdddd	Eeeee
14	Kdddd	Eeeee
15	Kdddd	Eeeee
16	Kdddd	Eeeee

The Masked Event Drum with Word Output features sixteen steps and sixteen 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 (Fffff field). 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).

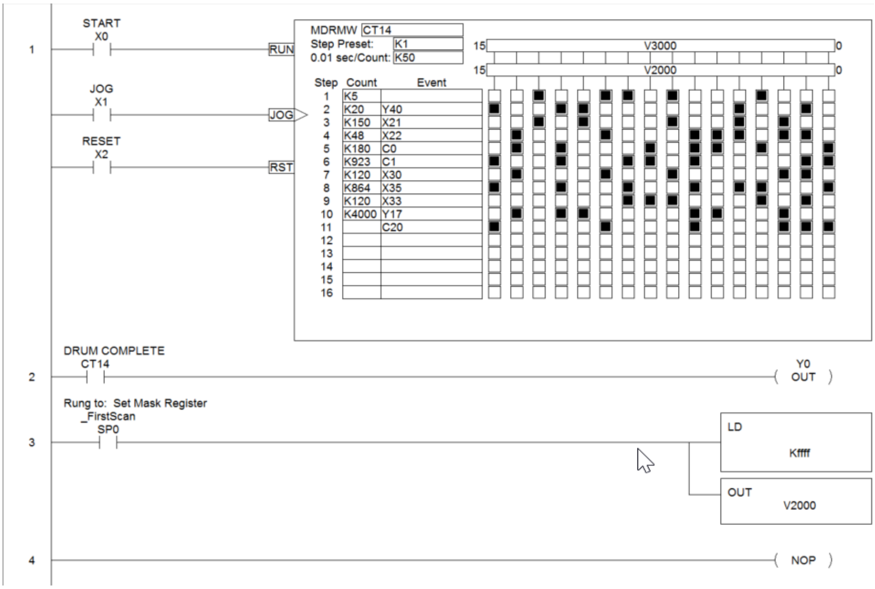
Whenever the Start input is energized, the drum's timer is enabled. As long as the event is true for the current step, the timer runs during that step. When the step count equals the counts per step, the drum transitions to the next step. This process stops when the last step is complete, or when the 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	CTaaa	–	0 – 377
Preset Step	Kbb	K	1 – 16
Timer base	Kcccc	K	0 – 99.99 seconds
Counts per step	Kdddd	K	0 – 9999
Event	Eeeee	X, Y, C, GX, GY, S, T, CT, CP	see memory map
Word Output	Fffff	V	see memory map
Output Mask	Ggggg	V	see memory map

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	Ranges of (n)	Function	Counter Bit Function
CTA(n)	0–374	Counts in step	CT(n) = Drum Complete
CTA(n+1)	1–375	Timer value	CT(n+1) = (not used)
CTA(n+2)	2–376	Preset Step	CT(n+2) = (not used)
CTA(n+3)	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 DirectSOFT. 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 V3000. The data bits in V2000 are logically ANDed with the output pattern of the current step in the drum, generating the contents of V3000. If you want all drum outputs to be off after power up, 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 10). A drum reset also resets CT14.



Step	Count	Event																
1	K5		□	□	■	□	□	■	■	□	■	□	□	□	■	□	□	□
2	K20	Y40	■	□	□	■	■	□	□	□	□	□	□	■	□	□	■	□
3	K150	X21	□	□	■	□	■	□	□	□	■	□	□	■	□	■	□	□
4	K48	X22	□	■	□	□	□	■	□	□	□	■	■	■	□	■	■	□
5	K180	C0	□	■	□	■	□	□	□	■	□	■	■	□	■	□	□	■
6	K923	C1	■	□	□	■	□	□	■	■	□	■	□	□	□	□	■	■
7	K120	X30	□	■	□	□	□	■	□	□	■	□	□	□	□	■	■	□
8	K864	X35	■	□	□	■	□	□	■	□	□	■	□	■	■	□	□	■
9	K120	X33	□	□	□	□	□	□	■	■	■	□	□	□	■	□	■	□
10	K4000	Y17	□	■	□	■	■	□	□	□	□	■	■	□	□	■	□	□
11		C20	■	□	□	□	□	■	□	□	□	■	□	□	□	■	■	■
12			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
13			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
14			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
15			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
16			□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□

NOTE: *The ladder program must load constants in V2000 through V2012 to cover all mask registers for the eleven steps used in this drum.*

RLLPLUS Stage Programming

Chapter 7

In This Chapter...

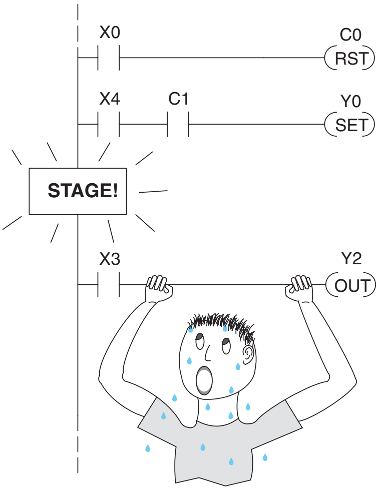
Introduction to Stage Programming

Stage Programming 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 RLLplus You won't have to discard any training or experience you already have. Stage programming simply allows you to divide and organize an 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.
- 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!

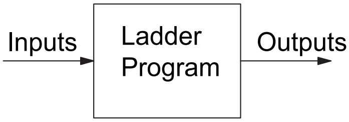
So, we encourage you to study stage programming and add it to your toolbox of programming techniques. This chapter is designed as a self-paced tutorial on stage programming. For best results:

- Start at the beginning and do not skip over any sections.
- Study each stage programming 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.

***NOTE:** Stage and relay ladder logic can both exist in the same program. The only restriction is that all stage logic must be below the regular relay ladder logic in the program.*

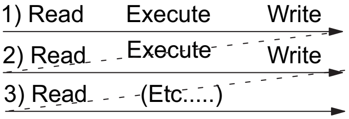
Learning to Draw State Transition Diagrams

Introduction to Process States



Those familiar with ladder program execution know that 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 just 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.

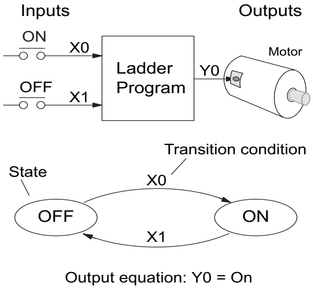
We can organize and divide ladder logic into sections called stages, representing process states. But before we describe stages in detail, we will reveal the secret to understanding stage programming: state transition diagrams.

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 just a tool to help us draw a picture of our process! You'll discover that if we can get the picture right, our program will also be right!

A 2–State Process

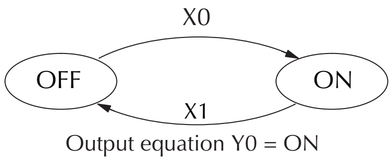
Consider the simple process shown to the right, which controls an industrial motor. We will use a green momentary SPST pushbutton to turn the motor on, and a red one to turn it off. The machine operator will press the appropriate pushbutton for just a second or so. The two states of our process are ON 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.

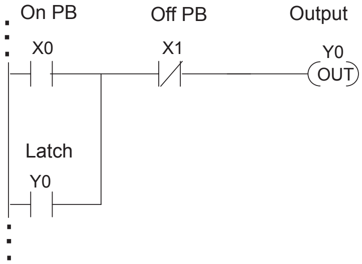
Next, we will implement the state diagram first as RLL, then as a stage program. This will help you see the relationship between the two methods in problem solving.



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.

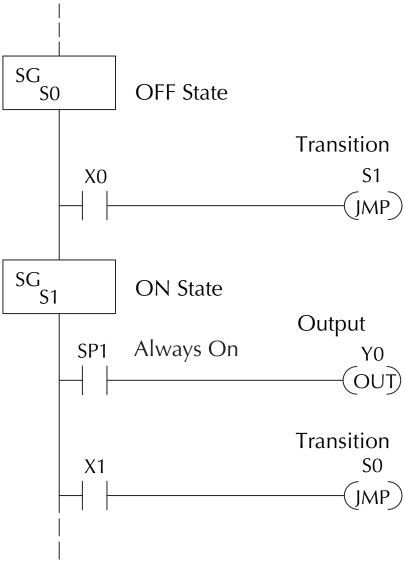
RLL Equivalent



The RLL solution is shown to the right. Output control relay, Y0, has a dual purpose. It turns the motor on and off and acts as a latching relay. 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 turns on the motor output Y0 which now has power flow and sets the latch Y0. It will remain on after the X0 contact opens.

When the Off pushbutton (X1) is pressed, it opens the normally-closed X1 contact, which turns off motor output Y0 and also resets the latch.

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.

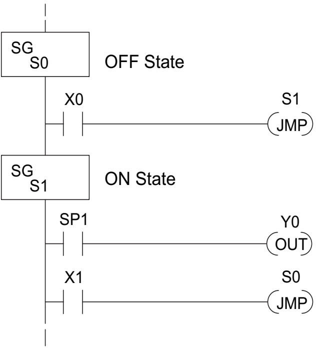
When the Off pushbutton (X1) is pressed, a transition back to the Off State occurs. The JMP S0 instruction executes, which simply turns off the Stage bit S1 and turns on Stage bit S0. On the next PLC scan, the CPU will not execute Stage S1, so the motor output Y0 will turn off. The Off state (Stage 0) will be ready for the next cycle.

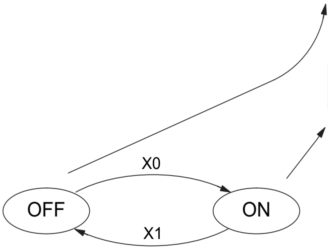
Let's Compare

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.

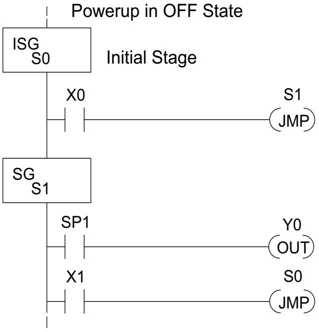
Now, we challenge anyone to easily identify the same states in the RLL program on the previous page!





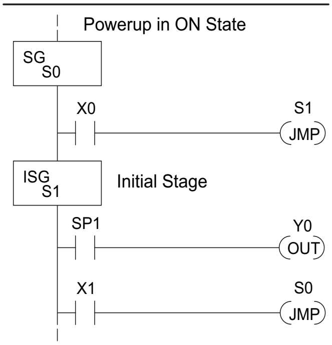
Initial Stages

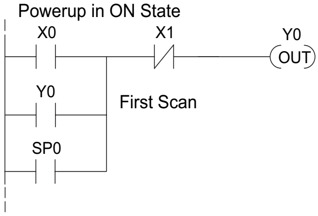
At power up and Program-to-Run Mode transitions, the PLC always begins with all normal stages (SG) off. So, the stage programs shown so far have actually had no way to get started (because rungs are not scanned unless their stage is active).



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!

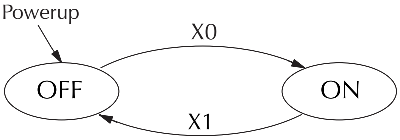
We can change both programs so that the motor is ON at powerup. In the RLL below, we must add a first scan relay SP0, latching Y0 on. In the stage example to the right, we simply make Stage S1 an initial stage (ISG) instead of S0.





***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 power up 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.

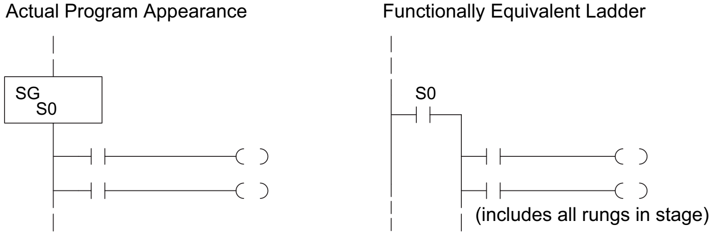


What Stage Bits Do

You may recall that a stage is just a section of ladder program which is either active or inactive at a given moment. All stage bits (S0 to 1777) reside in the PLC's image register as individual status bits. Each stage bit is either a boolean 0 or 1 at any time.

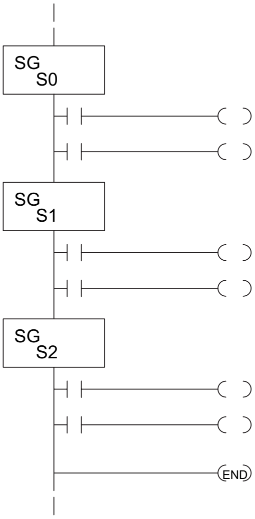
Program execution always reads ladder rungs from top to bottom, and from left to right. The drawing below shows the effect of stage bit status. The ladder rungs below the stage instruction continuing until the next stage instruction or the end of program belong to stage 0. Its equivalent operation is 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).


Actual Program Appearance
Functionally Equivalent Ladder
SG
S0
S0
(includes all rungs in stage)

Stage Instruction Characteristics

The inline stage boxes on the left power rail divide the ladder program rungs into stages. Some stage rules are:

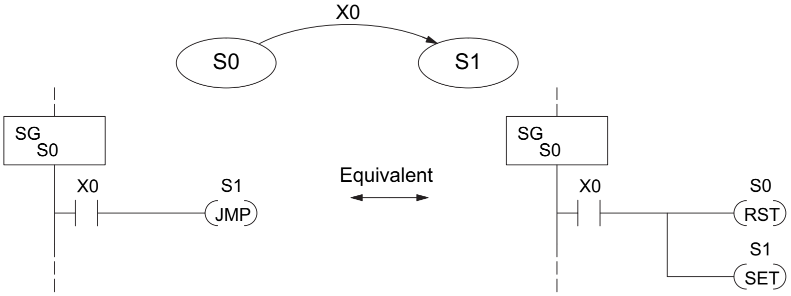


- Execution – Only logic in active stages are executed on any scan.
- Transitions – Stage transition instructions take effect on the next occurrence of the stages involved.
- Octal numbering – Stages are numbered in octal, like I/O points, etc. So “S8” is not valid.
- Total Stages – The D4-454 offers up to 1024 stages (S0 to 1777 in octal).
- 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

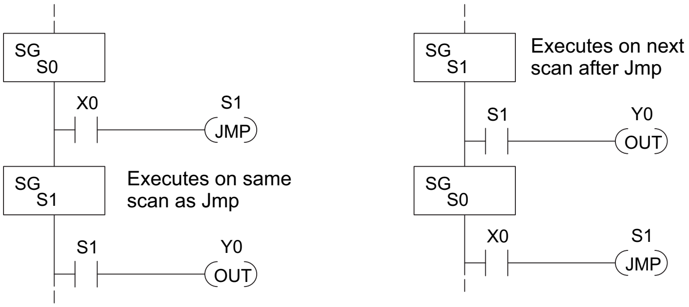
Stage Jump, Set, and Reset Instructions

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.

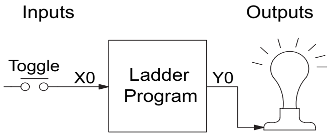


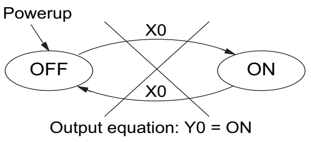
NOTE: *Assume we start with Stage 0 active and Stage 1 inactive for both examples.*

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 just 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.

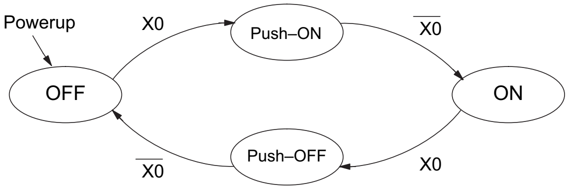




A typical first approach is to use X0 for both transitions (like the example shown to the right). However, this is incorrect (please keep reading).

Note that this example differs from the motor example, because now we have just one pushbutton. When we press the pushbutton, both transition conditions are met. We would just 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)!

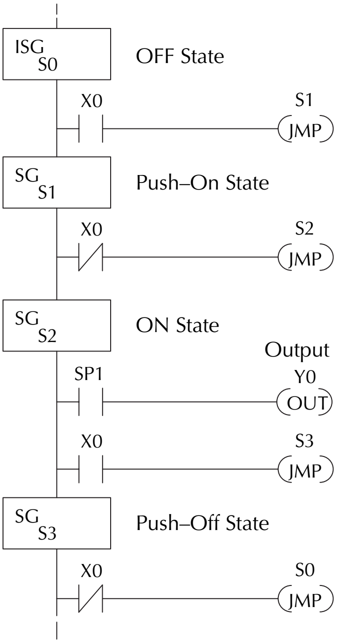
The solution is to make the push and the release of the pushbutton separate events. Refer to the new state transition diagram below. At power up we enter the OFF state. When switch X0 is pressed, we enter the Press-ON state. When it is released, we enter the ON state. Note that X0 with the bar above it denotes X0 NOT.



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 power up 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 power up 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 1024 total stages are available in the D4-454, numbered 0 to 1777 in octal.
- 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 power up.
- Place the outputs or actions in the appropriate stages.

You'll notice that Steps 1 through 3 just prepare us to write the stage program in Step 4. However, the program virtually writes itself because of the preparation beforehand. Soon you'll 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

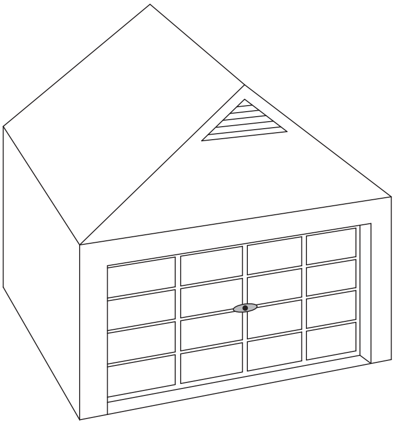
In this next stage programming example, we'll create a garage door opener controller. Hopefully, most readers are familiar with this application, and we can have fun, too!

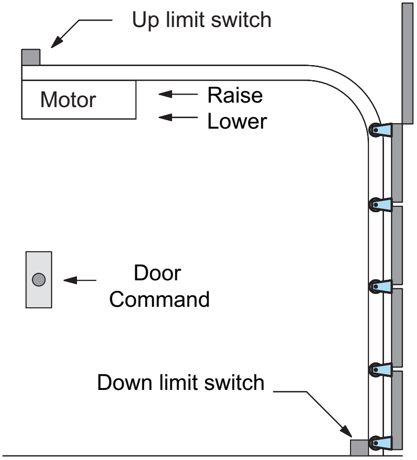
The first step we must take is to describe how the door opener works. We will start by achieving the basic operation, waiting to add extra features later. Stage programs are very easy to modify.

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 reach 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 just a simple pushbutton. Whether wall-mounted as shown, or a radio-remote control, all door control commands logical OR together as one pair of switch contacts.

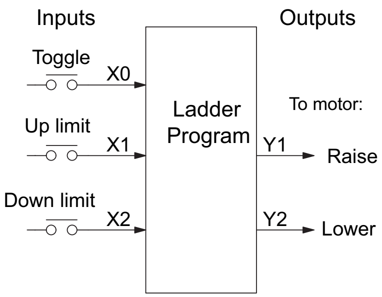




Draw the Block Diagram

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.

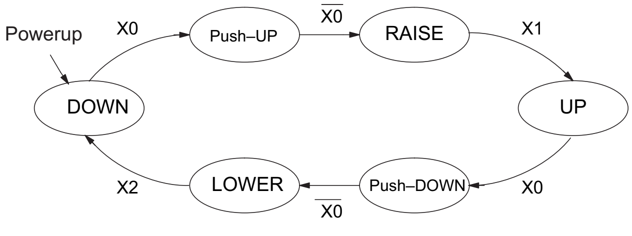
The controller has two outputs to drive the motor. Y1 is the up (raise the door) command, and Y2 is the down (lower the door) command.



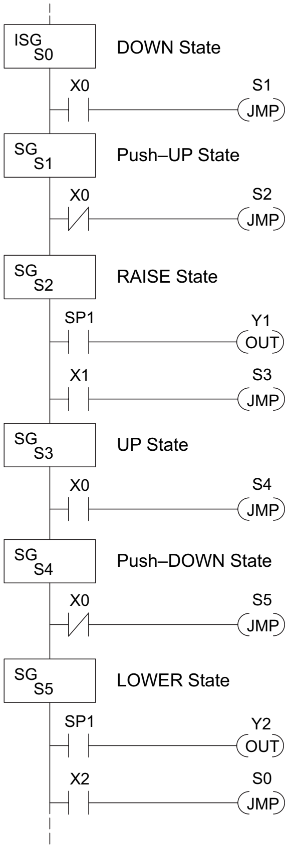
Draw the State Diagram

Now we are ready to draw the state transition diagram. Like the previous light bulb controller example, this application also has just 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 power up, so the desired power up 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.

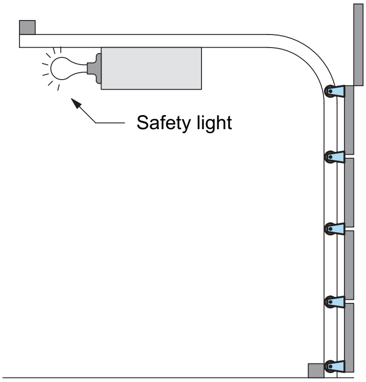
NOTE: *The only special thing about an initial stage (ISG) is that it is automatically active at power up. Afterwards, it is just like any other.*

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.

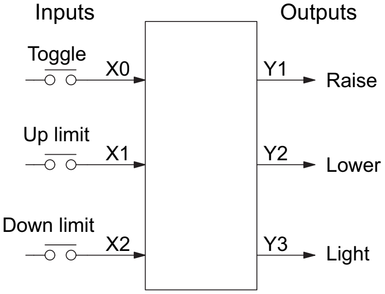
The safety light is standard on many commercially-available garage door openers. It is shown to the right, mounted on the motor housing. The light turns on upon any door activity, remaining on for approximately 3 minutes afterwards.

This part of the exercise will demonstrate the use of parallel states in our state diagram. Instead of using the JMP instruction, we'll use the set and reset commands.



Modify the Block Diagram and State Diagram

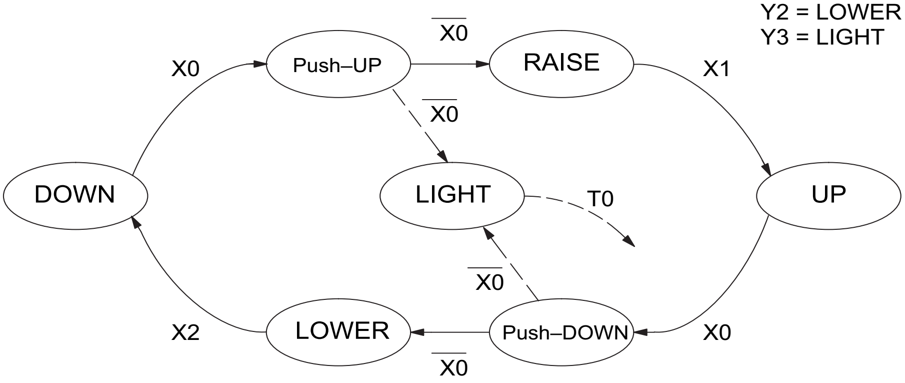
To control the light bulb, we add an output to our controller block diagram, shown to the right, Y3 is the light control output.



In the diagram below, we add an additional 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 just becomes inactive, and the light goes out.

Output equations: Y1 = RAISE
Y2 = LOWER
Y3 = LIGHT



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 closes, 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!

- Just 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.

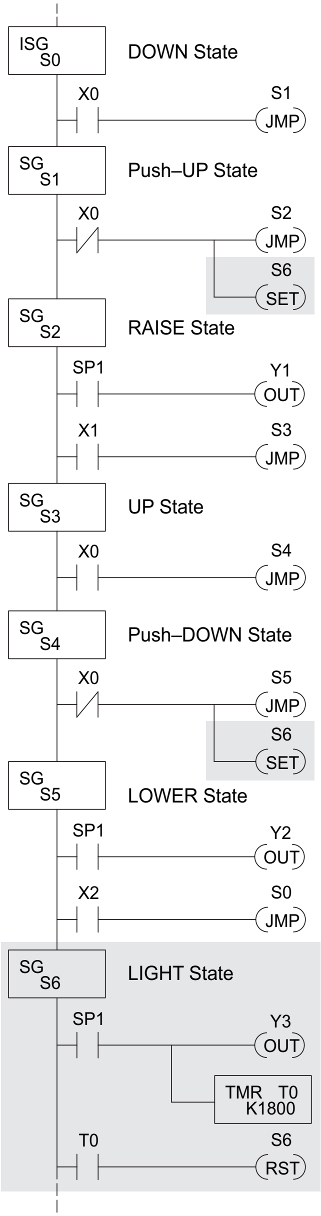
In stage S6, we turn on the safety light by energizing Y3. Special relay contact SP1 is always on. Timer T0 times at 0.1 second per count. To achieve 3 minutes time period, we calculate:

$$K = \frac{3 \text{ min.} \times 60 \text{ sec/min}}{0.1 \text{ sec/count}}$$

$$K = \quad 1800 \text{ counts}$$

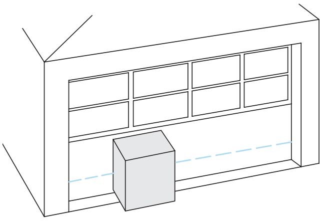
The timer has power flow whenever stage S6 is active. The corresponding timer bit T0 is set when the timer expires. So three minutes later, T0=1 and the instruction Reset S6 causes the stage to be inactive.

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.

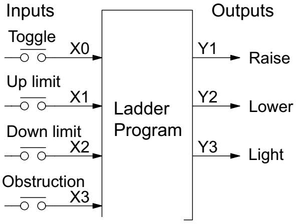


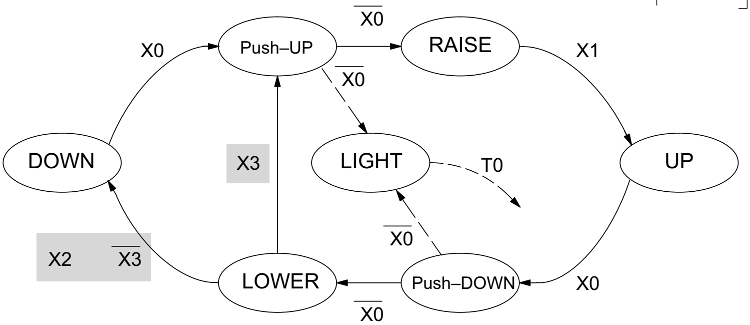
Add Obstruction Stop Feature

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.

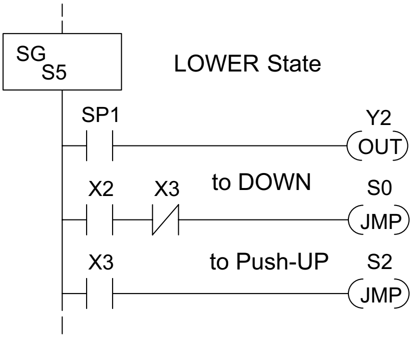




Exclusive Transitions

It is theoretically possible that 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.

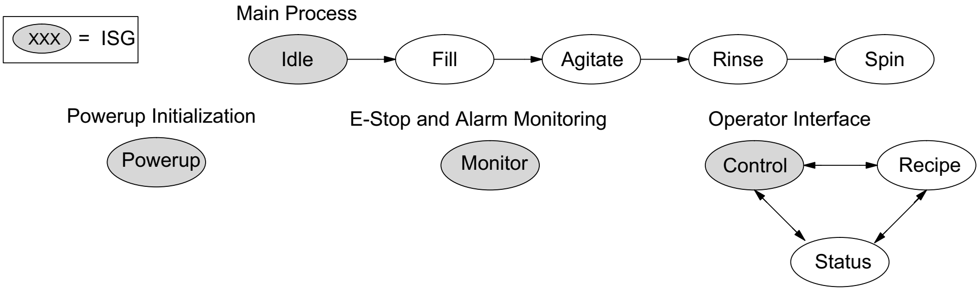


Stage Program Design Considerations

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, just put them in a stage that is always on.

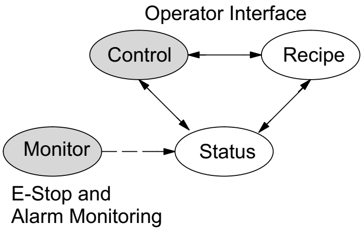
The following figure shows a typical application. During operation, the primary manufacturing activity Main Process, Power up Initialization, E-Stop and Alarm Monitoring, and Operator Interface are all running. At power up, four initial stages shown begin operation.



In a typical application, the separate stage sequences above operate as follows:

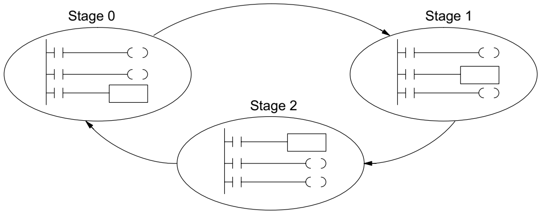
- Power up Initialization – This stage contains ladder rung tasks done just once at power up. Its last rung resets the stage, so this stage is only active for one scan (or only as many scans that are required).
- Main Process –This stage sequence controls the heart of the process or machine. One pass 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., independently 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 up 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 power up by using an initial stage type (ISG).



Most all instructions work just like they do in standard RLL. You can think of a stage just like a miniature RLL program which 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.
- 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. Therefore, use the OROUT instruction instead when you want multiple stages to have a logical OR control of an output, while the stage is active.

One-Shot or PD coils – Use care if you must use a Positive Differential coil in a stage. Remember that 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 which you want to do only once (on 1 scan), it can be placed in a stage which transitions to the next stage on the same scan.

Counter – In 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.

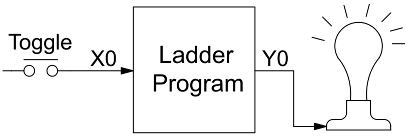
The ordinary Counter instruction does have a restriction inside stages: it may not be reset from other stages using the RST instruction for the counter bit. However, the special Stage counter provides a solution (see next paragraph).

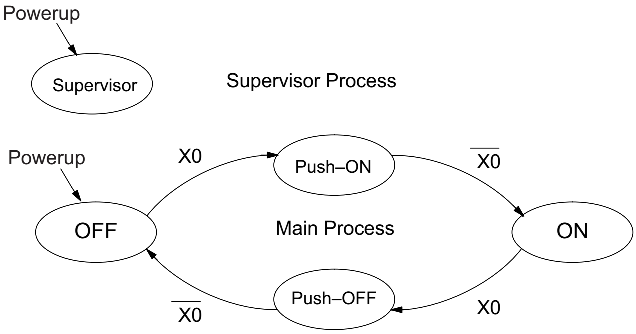
Stage Counter – The Stage Counter has the benefit that its count may be globally reset from other stages by using the RST instruction. It has a count input, but no reset input. This is the only difference from a standard counter.

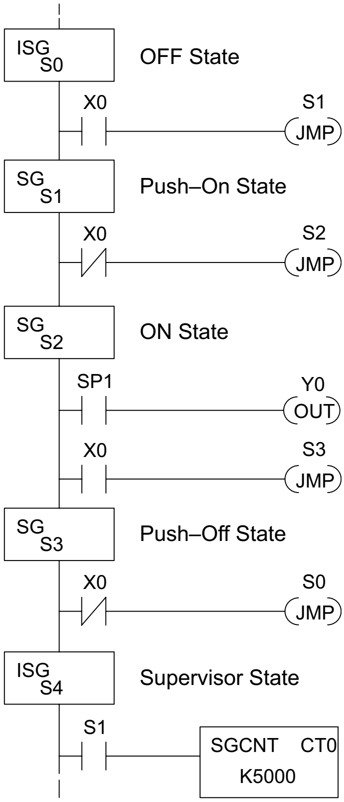
Drum – Realize that the drum sequencer is its own process, and is a different programming method than stage programming. If you need to use a drum with stages, be sure to place the drum instruction in an ISG stage that is always active.

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 which occurs. 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.*

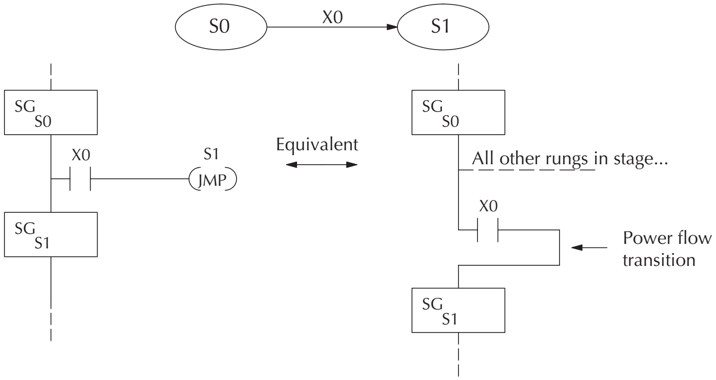
Stage Counter

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.

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 DirectSOFT, you may use the power flow method for stage transitions.

The main requirement is that 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.



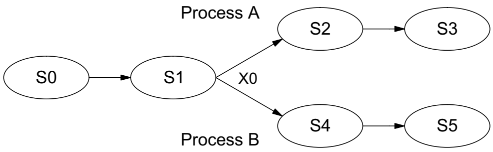
Remember that the Stage JMP instruction may occur anywhere in the current stage, and the result is the same. However, power flow transitions, as shown above, must occur as the last rung in a stage. All other rungs in the stage will precede it.

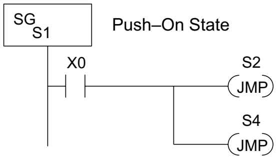
The power flow transition method does eliminate one Stage JMP instruction, its only advantage. However, it is not as easy to make program changes as using the Stage JMP. Therefore, we advise using Stage JMP transitions for most programmers.

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).

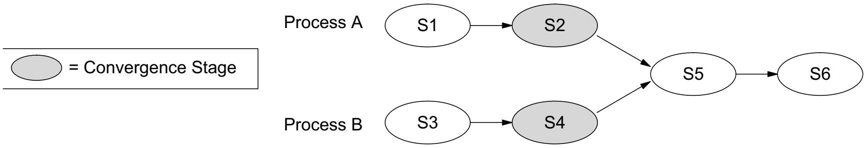




NOTE *that if we want Stages S2 and S4 to energize exactly on the same scan, both stages must be located below or above Stage S1 in the ladder program (see the explanation at the bottom of page 7–7). Overall, parallel branching is easy!*

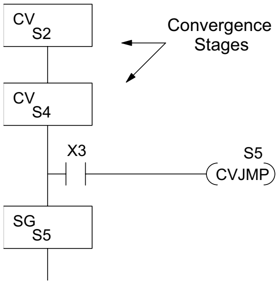
Converging Processes

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)

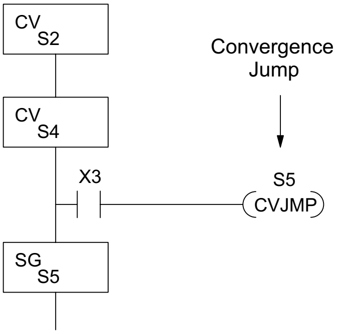
While the converging principle is simple enough, it brings a new complication. As parallel processing completes, the multiple processes almost never finish at the same time. In other words, how can we know whether Stage S2 or S4 will finish last? This is an important point, because we have to decide 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 together 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)

Remember, 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 Jump

Convergence Stage Guidelines

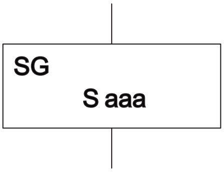
The following summarizes the requirements in the use of convergence stages, including some tips for their effective application:

- A convergence stage is to be used as the last stage of a process which is running in parallel to another process or processes. A transition to the convergence stage means that a particular process is finished and represents a waiting point until all other parallel processes also finish.
- The maximum number of convergence stages which make up one group is 16. In other words, a maximum of 16 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.

RLLPLUS (Stage) Instructions

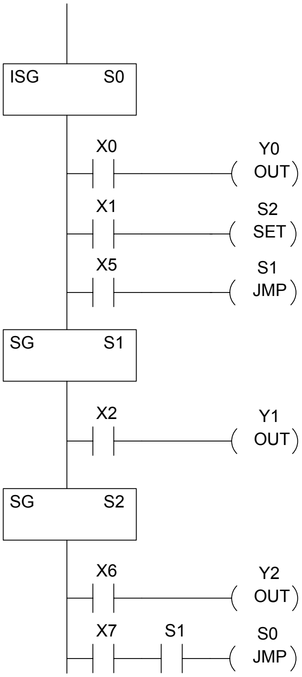
Stage (SG)

The Stage instructions are used to create structured RLLPLUS programs. Stages are program segments which 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.



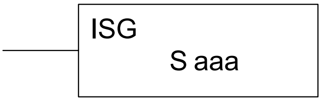
Operand Data Type	D4-454 Range
	aaa
Stage S	0–1777

The following example is a simple RLLPLUS program. This program utilizes an Initial Stage, Stage, and Jump instructions to create a structured program.



Initial Stage (ISG)

The Initial Stage instruction is normally used as the first segment of an RLLPLUS program. Multiple Initial Stages are allowed in a program. They 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.

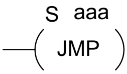


Operand Data Type	D4-454 Range
	aaa
Stage S	0–1777

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)

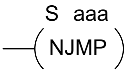
The Jump instruction allows the program to transition from an active stage containing the jump instruction to another stage (specified in the instruction). The jump occurs when the input logic is true. The active stage containing the Jump will deactivate 1 scan later.



Operand Data Type	D4-454 Range
	aaa
Stage S	0–1777

Not Jump (NJMP)

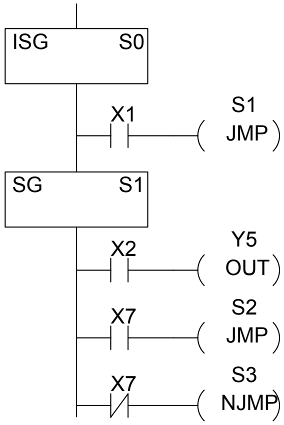
The Not Jump instruction allows the program to transition from an active stage which contains the jump instruction to another which is specified in the instruction. The jump will occur when the input logic is false. The active stage that contains the Not Jump will be deactivated 1 scan after the Not Jump instruction is executed.



Operand Data Type	D4-454 Range
	aaa
Stage S	0–1777

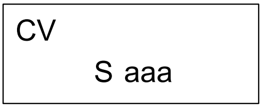
Not Jump (NJMP)

In the following example, only stage ISG S0 will be active when program execution begins. When X1 is on, program execution will jump from Initial Stage 0 to Stage 1.



Converge Stage (CV) and Converge Jump (CVJMP)

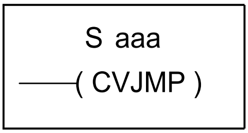
The Converge Stage instruction is used to group certain stages together by defining them as Converge Stages.



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.

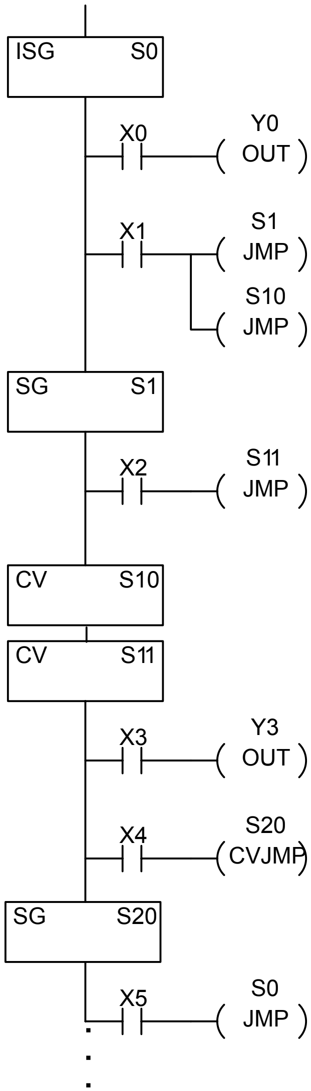
Additional logic instructions are only allowed following the last Converge Stage instruction and before the CVJMP instruction. Multiple CVJMP instructions are allowed.

Converge Stages must be programmed in the main body of the application program. This means they cannot be programmed in Subroutines or Interrupt Routines.

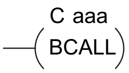


Operand Data Type	D4-454 Range
	aaa
Stage S	0–1777

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.



Block Call (BCALL)



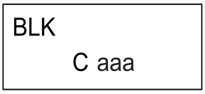
The stage block instructions are used to activate a block of stages. The Block Call, Block, and Block End instructions must be used together. The BCALL instruction is used to activate a stage block. There are several things you need to know about the BCALL instruction.

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 automatically be turned off. 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		D4-454 Range
		aaa
Control Relay	S	0–1777

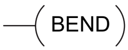
Block (BLK)



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 any where else in the program

Operand Data Type		D4-454 Range
		aaa
Control Relay	S	0–1777

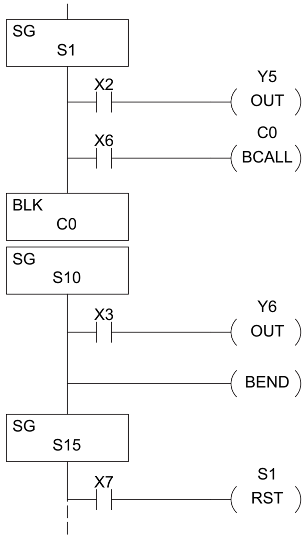
Block End (BEND)



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.

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.

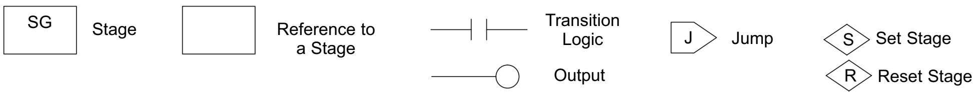
This allows the stages between S10 and the Block End instruction to operate as programmed. If the BCALL instruction is turned off, or if the stage containing the BCALL instruction is turned off, then all stages between the BLK and BEND instructions are automatically turned off.



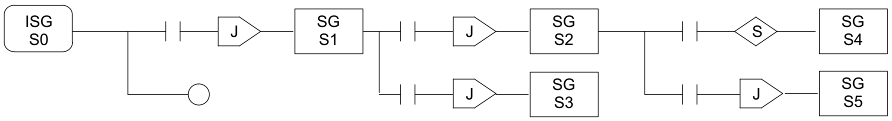
If you examine S15, you will notice that X7 could reset Stage S1, which would disable the BCALL, thus resetting all stages within the block.

Stage View in DirectSOFT

The Stage View option in DirectSOFT 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 up 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. What are Stage Bits?

A. A stage bit is just 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 power up.
- 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 power up.
- 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 Reset 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 DirectSOFT, we list them separately from two preceding questions.

Q. Can I have a stage which is active for only one scan?

A. Yes, but this is not the intended use for a stage. Instead, just make a ladder rung active for 1 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 power up. Afterwards, it works just like any other stage. You can have multiple initial stages, if required. Use an initial stage for ladder that must always be active, or as a starting point.

Q. Can I 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 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

Chapter 8

In This Chapter...

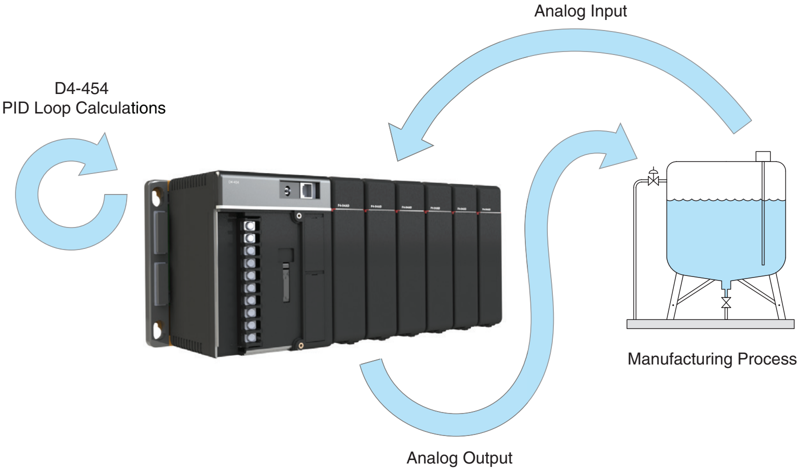
D4-454 PID Control

D4-454 PID Control Features

Along with control functions discussed in this manual, the D4-454 PLC features PID process control capability. The D4-454 PID process control loops offer the same features offered in much larger PLCs. The primary features are:

- Up to 16 PID 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 D4-454 CPU has process control loop capability in addition to ladder program execution. You can select and configure up to sixteen loops. All sensor and actuator wiring connects directly to DL405 analog modules. All process variables, gain values, alarm levels, etc., associated with each loop reside in a Loop Variable Table in the CPU. The D4-454 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.



DirectSOFT programming software, release 6.1, or later, is used for configuring analog control loops in the D4-454. DirectSOFT uses dialog boxes to help you set up the individual loops. After completing the setup, you can use DirectSOFT’s PID Trend View to tune each loop. The configuration and tuning selections you make are stored in the D4-454’s V-memory , which can be set as retentive. The loop parameters also may be saved to disk for recall later.

PID Loop Feature	Specifications
Number of loops	Selectable, 16 maximum
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 0.1 seconds for 5 to 8 loops 0.2 seconds for 9 to 16 loops
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 0.0 to 99.99
Integrator (Reset)	Specify reset time of 0.0 to 99.99 in units of seconds or minutes
Derivative (Rate)	Specify the derivative time from 0.0 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?

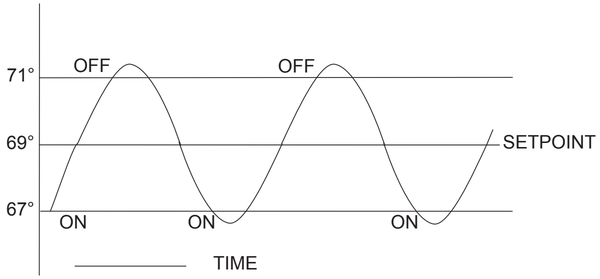
In this discussion, we will explain why PID control is used in process control instead of trying to provide control by simply using an analog input and a discrete output.

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°, the furnace will be turned on to provide heat at, normally, 2° below the setpoint. In this case, it would turn on at 67°. When the temperature reaches 71°, 2° 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°, the A/C unit will turn on when the sensed temperature reaches 2° above the setpoint or 78°, and turn off when the temperature reaches 74°. 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 70mph, the setpoint. Now, the car is cruising at a steady 70mph 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 70mph 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 70mph 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 70mph 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 70mph, 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 70mph. 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 D4-454 PID Control

The D4-454 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 D4-454 PLC has the ability to directly accept signals from electronic sensors, such as thermocouples, pressure transducers, VFDs, etc. These signals may be used in mathematically derived control systems.

In addition, the D4-454 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, pressure, 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 DL405 PID control provides feedback loops using the PID algorithm. The control output is computed from the measured process variable as follows:

Let:

- Kc = proportional gain
- Ti = Reset or integral time
- Td = 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”

$$M(t) = Kc\ [\ e(t) + 1/Ti \int 0t\ e(x)\ dx + Td\ d/dt\ e(t)\ ] + Mo$$

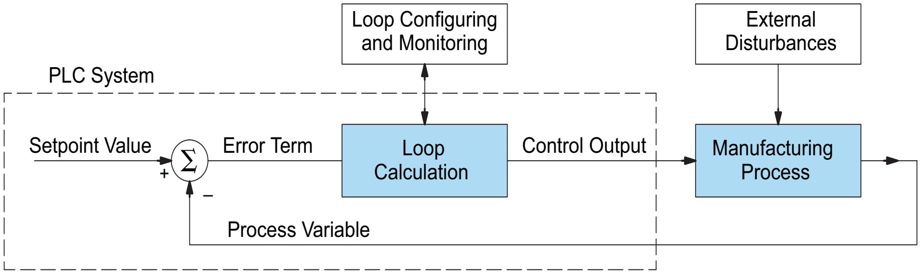
The analog input module receives the process variable in analog form along with an operator entered setpoint; the CPU computes the error. The error is used in the algorithm computation to provide corrective action at the control output. The function of the control action is based on an output control, 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 DL405 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 4-20mA current output module to control a valve.

With DirectSOFT, 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 – This is 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 what 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 – This is 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 D4-454 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 D4-454 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 D4-454 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
  PVn = Process variable at nth sample
  en = SP – PVn = Error at nth sample
  Mo = Value to which the controller output has been initialized

- Then:

  M_n = Control output at n^{th} sample

$$Mn = Kc * en + Ki \sum ni=1 \; ei + Kr \, (en - en\text{-}1) + Mo$$

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:

$$Mn = Kc * en + Ki \sum ni=1 \; ei + Kr \, (PVn - PVn\text{-}1) + Mo$$

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 D4-454 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:

$$Mxo = Mo$$

$$Mx = Ki * en + Mxn-1$$

$$Mn = Kc * en - Kr(PVn-PVn-1) + Mxn$$

The D4-454 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 D4-454 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 text books 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:

$$Mx = Ki * en + Mxn-1$$

For example, assume the output is controlling a valve and the PV remains at some value greater than the setpoint. The negative error (en) 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 D4-454 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 D4-454 CPU is programmed to solve the overshoot problem by either freezing the bias term, or by 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.

$$Mx = Ki * en + Mxn-1$$

$$M = Kc * en - Kr(PVn - PVn-1) + Mx$$

$Mn = 0 \quad \text{if } M < 0$
$Mn = M \quad \text{if } 0 \leq M \leq 1$
$Mn = 1 \quad \text{if } M > 1$

$Mxn = Mx \quad \text{if } 0 \leq M \leq 1$
$Mxn = Mxn-1 \quad \text{otherwise}$

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.

$$Mx = Ki * en + Mxn-1$$

$$M = Kc * en - Kr(PVn - PVn-1) + Mx$$

$Mn = 0 \quad \text{if } M < 0$
$Mn = M \quad \text{if } 0 \leq M \leq 1$
$Mn = 1 \quad \text{if } M > 1$

$Mxn = Mx \quad \text{if } 0 \leq M \leq 1$
$Mxn = Mn - Kc * en - Kr(PVn - PVn-1) \quad \text{otherwise}$

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.

$Mx = Mx * SPn / SPn-1$ if the loop is direct acting

$Mx = Mx * SPn-1 / SPn$ if the loop is reverse acting

$Mxn = 0$ if $Mx < 0$
$Mxn = Mx$ if $0 \leq Mx \leq 1$
$Mxn = 1$ 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 $Kc \neq 0$

$Ki = Ts / Ti$ if $Kc = 0$ (I or ID only)

$Kr = Kc * (Td / Ts)$ if $Kc \neq 0$

$Kr = Td / Ts$ 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:

$$\Delta Mn = M - Mn-1$$

$$\Delta Mn = Kc * (en - en-1) + Ki * (PVn - 2 * PVn-1 + PVn-2)$$

Bumpless Transfer

The D4-454 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	Velocity PID Algorithm
SP = PV	SP = PV
Mx = M	

The bumpless transfer feature of the D4-454 is available in two types: Bumpless I and Bumpless II (see page 8-27). The transfer type is selected when the loop is set up.

Loop Alarms

The D4-454 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 DirectSOFT 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 D4-454 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 D4-454 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 D4-454 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.

$$Mx = -Ki * en + Mxn-1$$

$$M = -Kc * en + Kr(PVn-PVn-1) + Mxn$$

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:

$$en = (SP - PVn) * ABS(SP - PVn)$$

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.

en = 0 "SP - Deadband_Below_SP < PV < SP - Deadband_Above_SP"
en = P - PVn "otherwise."

The error will be squared first if both Error Squared and Error Deadband is 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, Yn, as shown below.

Position Algorithm

$$Y_n = Y_{n-1} + \frac{Ts}{Ts + \left(\frac{Td}{Kd}\right)} * (PV_n - Y_{n-1})$$

$$Mx = Ki * en + Mxn\text{-}1$$

$$M = Kc * en - Kr * (Yn\text{-}Yn\text{-}1) + Mx$$

Velocity Algorithm

$$\Delta M = Kc * (en - en\text{-}1) + Ki * en - Kr * (Yn - 2 * Yn\text{-}1 + Yn\text{-}2)$$

Ten Steps to Successful Process Control

Controllers such as the D4-454 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., which 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.

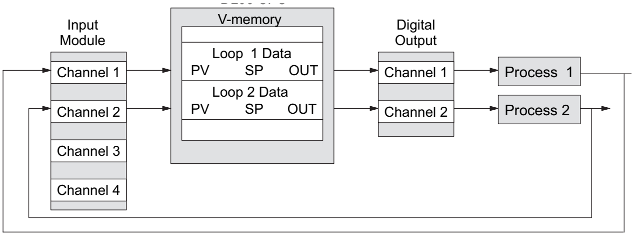
Step 3: Size and Scale Loop Components

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.) which matches the size of the load. An oversized actuator will have an overwhelming effect on your process after a SP change. However, an undersized actuator will allow the PV to lag or drift away from the SP after a 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 5 times that resolution (at LSB level). However, an over-sensitive sensor can cause control oscillations, etc. The D4-454 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. 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.



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. The most common wiring errors when installing PID loop controls are:
- Reversing the polarity of sensor or actuator wiring connections.
- Incorrect signal ground connections between loop components.

Step 6: Loop Parameters

After wiring and installation, choose the loop setup parameters. The easiest method for programming the loop tables is using DirectSOFT (6.1 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 8–42) 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 8–49). 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 setup parameters to disk.

PID Loop Setup

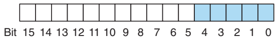
Some Things to Do and Know Before Starting

Have your analog module installed and operational before beginning the loop setup (refer to the D4-ANLG-M Manual). The D4-454 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 setup 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	V1200 – V7340 V10000-V17740	write
V7641	Number of Loops	BCD	1 – 16	write
V7642	Loop Error Flags	BITS	0 or 1	read

NOTE: *It is recommended to set the V-memory used to store the PID Setup Parameters to retentive. If power is removed from the CPU, these parameters will not be written to zero on power up.*

PID Error Flags, V7642



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.

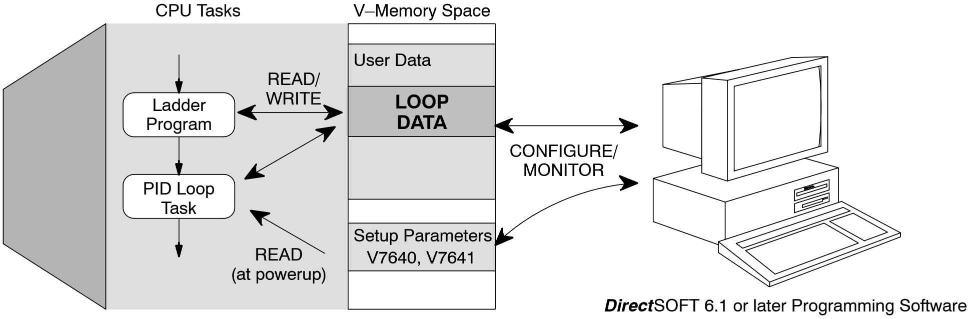
If you use the DirectSOFT loop setup dialog box, its automatic range checking prohibits possible setup errors. However, the setup parameters may be written using other methods such as RLL, so the error flag register may be helpful in those cases. The following table lists the errors reported in V7642.

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 16.
3	The loop table extends past (straddles) the boundary at V7377. Use an address closer to V1400.
4	The loop table extends past (straddles) the boundary at V36777. Use an address closer to V10000.

As a quick check, if the CPU is in Run mode and V7642=0000, there are no programming errors.

Establishing the Loop Table Size and Location

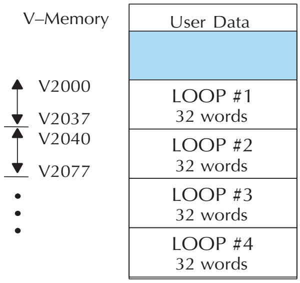
On a PROGRAM-to-RUN mode transition, the CPU reads the loop setup parameters as pictured below. At that moment, the CPU learns the location of the loop table and the number of loops it configures. Then during the ladder program scan, the PID Loop task uses the loop data to perform calculations, generate alarms, and so on. There are some loop table parameters the CPU will read or write on every loop calculation.



***NOTE:** The D4-454 CPU's PID algorithm requires DirectSOFT Version 6.1 (or later) and firmware version 1.0 (or later). 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.

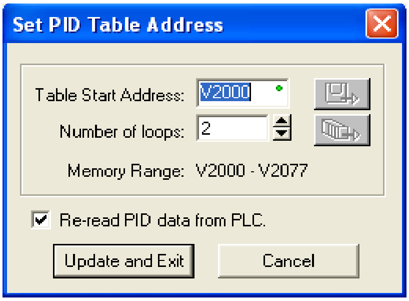


Determine the block of V-memory to be used for each PID loop. Besides being the beginning of the PID parameter memory block, the first address will be the start of loop 1 parameters. Remember, there are 32 words (0 to 37 octal) needed for each loop. Once you have determined the beginning V-memory address to be used, you can setup and store the PID parameters either directly in your RLL program or by the using PID Setup in DirectSOFT.

***NOTE:** Whether one or more loops are being setup, 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 DirectSOFT is the simplest way to setup the parameters. To setup the PID parameters, the D4-454 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 DirectSOFT. This can be seen in the dialog box below. After the address has been entered, the memory range will appear. Also, entering the number of PID loops (1 to 16) 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.



Set PID Table Address

Table Start Address: V2000

Number of loops: 2

Memory Range: V2000 - V2077

☑ Re-read PID data from PLC.

***NOTE:** Have an edited program open, then click on PLC > Setup > PID to access the Setup PID dialog.*

Loop Table Word Definitions

These are the loop parameters associated with each of the loops available in the D4-454. 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	Bits	Yes
3	Addr + 2	Setpoint Value (SP)	Word/Binary	Yes
4	Addr + 3	Process Variable (PV)	Word/Binary	Yes
5	Addr + 4	Bias (Integrator) Value	Word/Binary	Yes
6	Addr + 5	Control Output Value	Word/Binary	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	Word/Bcd	Yes
10	Addr + 11	Reset (Integral) Time Setting	Word/Bcd	Yes
11	Addr + 12	Rate (Derivative) Time Setting	Word/Bcd	Yes
12	Addr + 13	PV Value, Low-low Alarm	Word/Binary	No*
13	Addr + 14	PV Value, Low Alarm	Word/Binary	No*
14	Addr + 15	PV Value, High Alarm	Word/Binary	No*
15	Addr + 16	PV Value, High-high Alarm	Word/Binary	No*
16	Addr + 17	PV Value, deviation alarm (YELLOW)	Word/Binary	No*
17	Addr + 20	PV Value, deviation alarm (RED)	Word/Binary	No*
18	Addr + 21	PV Value, rate-of-change alarm	Word/Binary	No*
19	Addr + 22	PV Value, alarm hysteresis setting	Word/Binary	No*
20	Addr + 23	PV Value, error deadband setting	Word/Binary	Yes
21	Addr + 24	PV low-pass filter constant	Word/Bcd	Yes
22	Addr + 25	Loop derivative gain limiting factor setting	Word/Bcd	No**
23	Addr + 26	SP value lower limit setting	Word/Binary	Yes
24	Addr + 27	SP value upper limit setting	Word/Binary	Yes
25	Addr + 30	Control output value lower limit setting	Word/Binary	No**
26	Addr + 31	Control output value upper limit setting	Word/Binary	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	Word/Hex	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	–	0 –> 1 request
1	Automatic Mode Loop Operation request	Write	–	0 –> 1 request
2	Cascade Mode Loop Operation request	Write	–	0 –> 1 request
3	Bumpless Transfer select	Write	Mode I	Mode II
4	Direct or Reverse-Acting Loop select	Write	Direct	Reverse
5	Position / Velocity Algorithm select	Write	Position	Velocity
6	PV Linear / Square Root Extract select	Write	Linear	Sq. root
7	Error Term Linear / Squared select	Write	Linear	Squared
8	Error Deadband enable	Write	Disable	Enable
9	Derivative Gain Limit select	Write	Off	On
10	Bias (Integrator) Freeze select	Write	Off	On
11	Ramp/Soak Operation select	Write	Off	On
12	PV Alarm Monitor select	Write	Off	On
13	PV Deviation alarm select	Write	Off	On
14	PV rate-of-change alarm select	Write	Off	On
15	Loop mode is independent from CPU mode when set	Write	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 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)	Write	12 Bit	15 Bit
2	Analog Input filter	Write	Off	O
3	SP Input limit enable	Write	Disable	Enable
4	Integral Gain (Reset) units select	Write	Seconds	Minutes
5	Select Auto tune PID algorithm	Write	Closed loop	Open loop
6	Auto tune selection	Write	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 2 and 3)	Write	Not 16 bit	Select 16 bit
10	Select separate data format for input and output (See Notes 3 and 4)	Write	Same format	Separate formats
11	Control Output Range Unipolar/Bipolar select (See Notes 3 and 43)	Write	Unipolar	Bipolar
12	Output Data Format select (See Notes 3 and 4)	Write	12 Bit	15 Bit
13	Output data format 16-bit select (See Notes 3 and 4)	Write	Not 16 bit	Select16 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	Read	–	Auto
2	Cascade Mode Indication	Read	–	Cascade
3	PV Input LOW–LOW Alarm	Read	Off	On
4	PV Input LOW Alarm	Read	Off	On
5	PV Input HIGH Alarm	Read	Off	On
6	PV Input HIGH–HIGH Alarm	Read	Off	On
7	PV Input YELLOW Deviation Alarm	Read	Off	On
8	PV Input RED Deviation Alarm	Read	Off	On
9	PV Input Rate-of-Change Alarm	Read	Off	On
10	Alarm Value Programming Error	Read	–	Error
11	Loop Calculation Overflow/Underflow	Read	–	Error
12	Loop in Auto-Tune indication	Read	Off	On
13	Auto-Tune error indication	Read	–	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	–	0 –> Start
1	Hold Ramp / Soak Profile	Write	–	0 –> Hold
2	Resume Ramp / soak Profile	Write	–	0 –> Resume
3	Jog Ramp / Soak Profile	Write	–	0 –> Jog
4	Ramp / Soak Profile Complete	Read	–	Complete
5	PV Input Ramp / Soak Deviation	Read	Off	On
6	Ramp / Soak Profile in Hold	Read	Off	On
7	Reserved	Read	–	–
8–15	Current Step in R/S Profile	Read	Decode as byte (hex)	

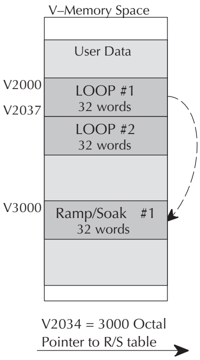
Bits 8–15 must be read as a byte to indicate the current segment number of the Ramp/ Soak generator in the profile. This byte will have the values 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, and 10, which represent segments 1 to 16 respectively. If the byte=0, then the Ramp/ Soak table is not active.

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 DirectSOFT 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

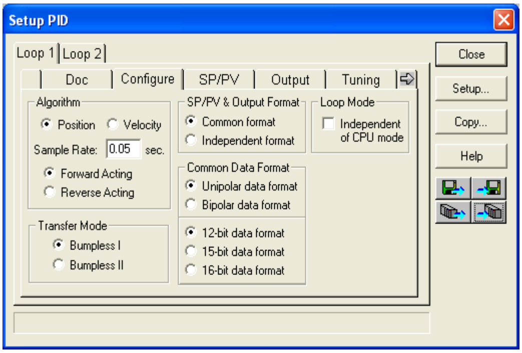
Ramp/Soak Table Programming Error Flags (Addr + 35)

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	–	–	–

Configure the PID Loop

Once the PID table is established in V-memory, configuring the PID loop continues with the DirectSOFT PID setup configuration dialog. You will need to check and fill in the data required to control the PID loop. Select Configure and the following dialog will appear for this process.



Setup PID

Loop 1 | Loop 2

Doc | Configure | SP/PV | Output | Tuning

Algorithm
- ● Position ○ Velocity
- Sample Rate: 0.05 sec.
- ● Forward Acting
- ○ Reverse Acting

Transfer Mode
- ● Bumpless I
- ○ Bumpless II

SP/PV & Output Format
- ● Common format
- ○ Independent format

Common Data Format
- ● Unipolar data format
- ○ Bipolar data format
- ● 12-bit data format
- ○ 15-bit data format
- ○ 16-bit data format

Loop Mode
- □ Independent of CPU mode

Select the Algorithm Type

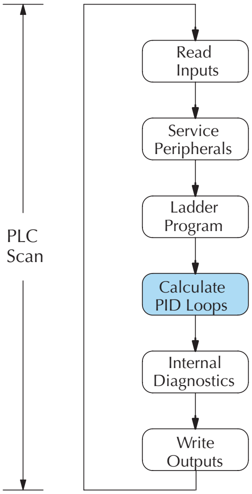
Chose either Position or Velocity. The default algorithm is Position. This is the choice for most applications which include heating and cooling loops as well as most position and level control loops. A typical velocity control will consist of a process variable such as a flow totalizer in a flow control 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 DL405 CPU, 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 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).

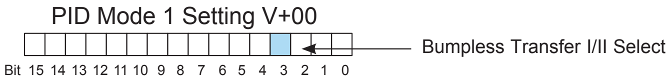
A reverse acting control loop is one where an increase in the control output results in a decrease in the PV. A common example of this would be a refrigeration system, where an increase in the cooling input causes a decrease in the PV (temperature).

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 Bumpless II.

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


PID Mode 1 Setting V+00
Bumpless Transfer I/II Select
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

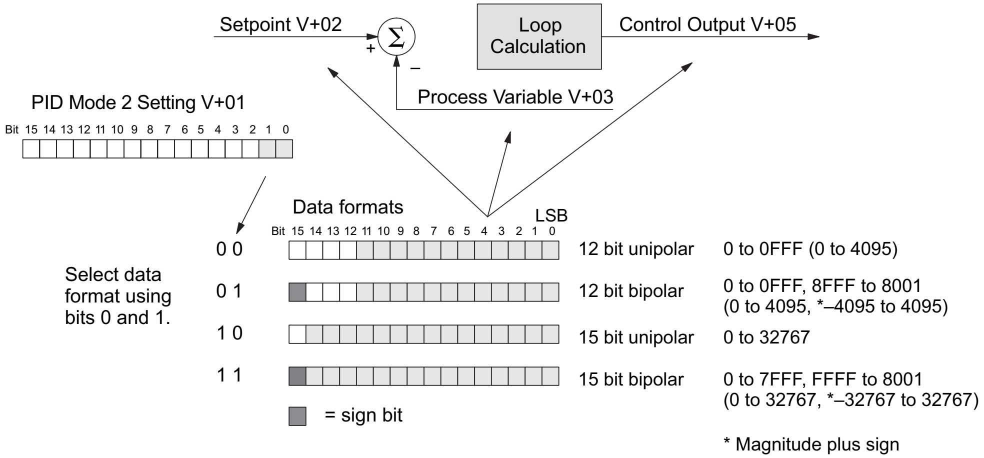
The transfer type can also be selected in an RLL program by setting bit 3 of PID Mode 1, V+00 setting as shown.

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.



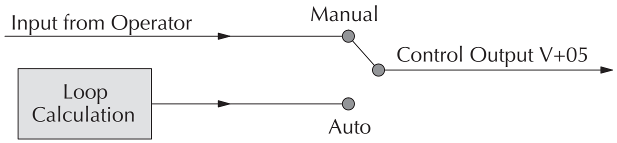
The data format determines the numerical interface between the PID loop and the PV sensor and the control output device. This selects the data format for both the SP and the PV.

Loop Mode

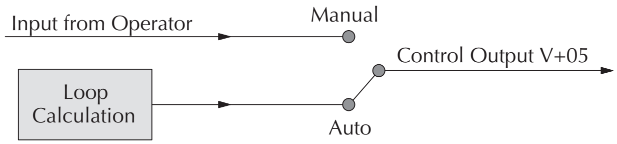
Loop Mode is a special feature that allows the PID loop controller to perform closed-loop control while the CPU is in the Program Mode. Careful thought must be taken before using this feature called Independent of CPU mode in the dialog. Before continuing with the PID setup, a knowledge of the three PID loop modes will be helpful.

The DL405 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.

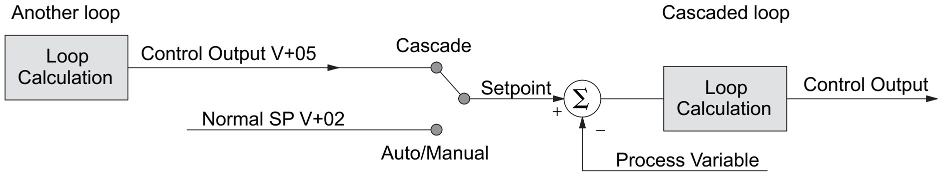
In Manual Mode, the loop is not executing PID calculations (however, loop alarms are still active). With regard to the loop table, the CPU stops writing values to location V+05 (control output) for that loop. It is expected that an operator or other intelligent source is manually controlling the output 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.



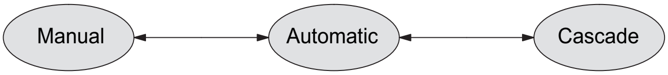
In Automatic Mode, the loop operates normally and generates new control output values. It calculates the PID equation and writes the result in location V+05 every sample period of that loop. The equivalent schematic diagram is shown below.



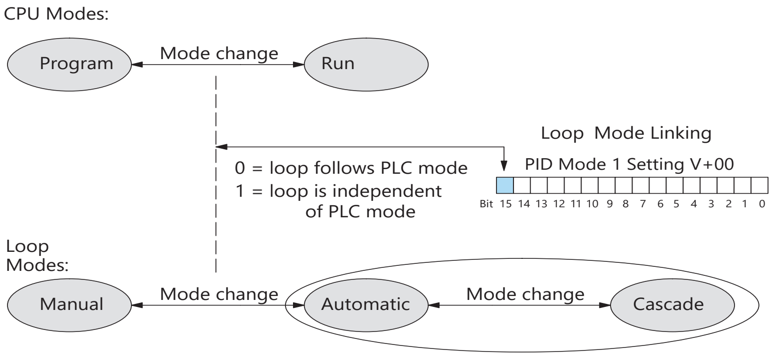
In Cascade Mode, the loop operates as it does in Automatic Mode, with one important difference. The data source for the SP changes from its normal location at V+02 to using the control output value, V+05, from another loop. So in Auto or Manual modes, the loop calculation uses the data at V+02. In Cascade 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.



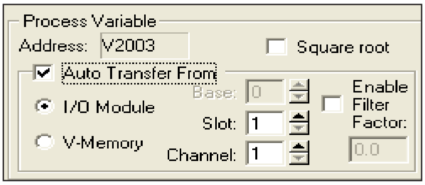
If bit 15 is set to one, then the loops will run independently of the CPU mode. It is like having two independent processors in the CPU... one is running the RLL program and the other is running the process loops.

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 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 setup 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.



Process Variable

Address: V2003 □ Square root

☑ Auto Transfer From

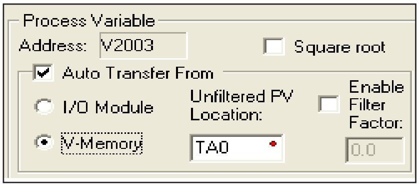
● I/O Module ○ V-Memory

Base: 0 Slot: 1 Channel: 1

□ Enable Filter Factor: 0.0

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.



Process Variable

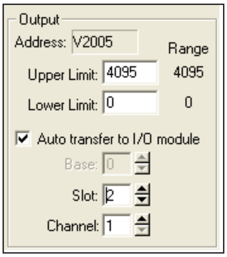
Address: V2003 □ Square root

☑ Auto Transfer From

○ I/O Module ● V-Memory

Unfiltered PV Location: TA0

□ Enable Filter Factor: 0.0



Output

Address: V2005 Range

Upper Limit: 4095 4095

Lower Limit: 0 0

☑ Auto transfer to I/O module

Base: 0

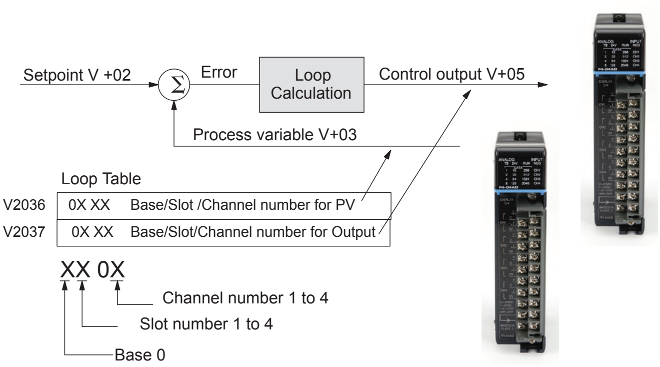
Slot: 2

Channel: 1

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 DirectSOFT. After you change the loop's parameter settings,*
restore bit 15 to 1.

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.



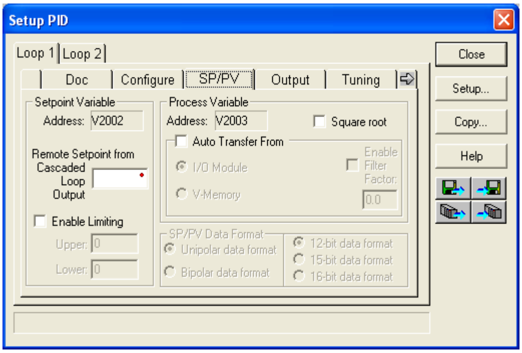
V2036	0X XX	Base/Slot /Channel number for PV
V2037	0X XX	Base/Slot/Channel number for Output

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 1 of the analog input. 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 setup 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.



Setpoint Variable
Address: V2002
Remote Setpoint from Cascaded Loop Output:
□ Enable Limiting
Upper: 0
Lower: 0

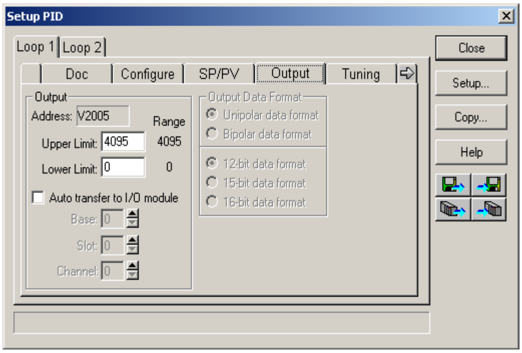
Process Variable
Address: V2003
□ Square root
□ Auto Transfer From
● I/O Module
○ V-Memory
□ Enable Filter Factor: 0.0

SP/PV Data Format
● Unipolar data format
○ Bipolar data format
● 12-bit data format
○ 15-bit data format
○ 16-bit data format

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.



Output
Address: V2005
Range
Upper Limit: 4095 — 4095
Lower Limit: 0 — 0
☐ Auto transfer to I/O module
Base: 0
Slot: 0
Channel: 0

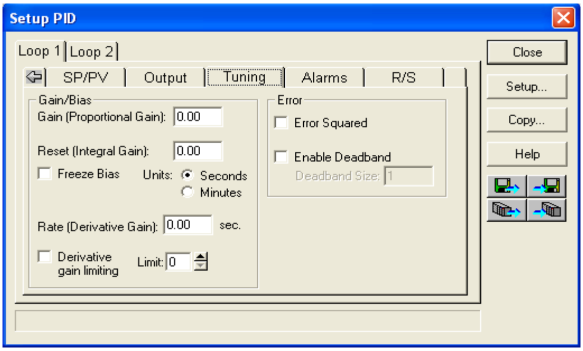
Output Data Format
● Unipolar data format
○ Bipolar data format
● 12-bit data format
○ 15-bit data format
○ 16-bit data format

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. The default value is zero, so this value MUST be changed.

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).



Setup PID

Loop 1 | Loop 2

SP/PV | Output | Tuning | Alarms | R/S

Gain/Bias

Gain (Proportional Gain): 0.00

Reset (Integral Gain): 0.00

□ Freeze Bias Units: ● Seconds ○ Minutes

Rate (Derivative Gain): 0.00 sec.

□ Derivative gain limiting Limit: 0

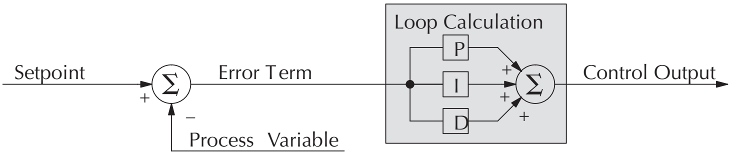
Error

□ Error Squared

□ Enable Deadband

Deadband Size: 1

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.

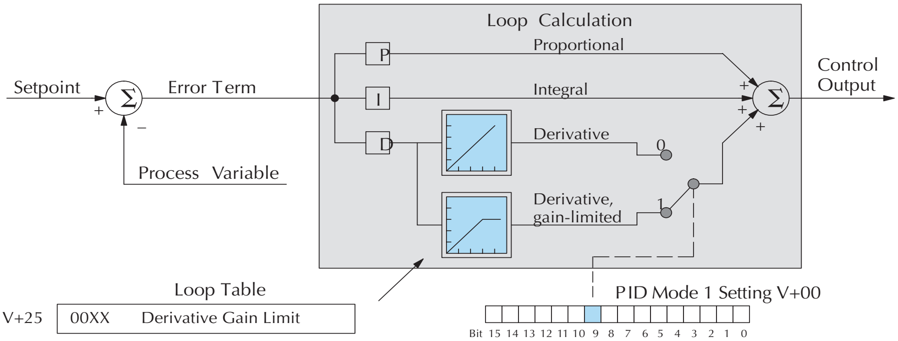
Reset (Integral Gain) – Values range from 0001 to 9998, but they are used internally as xx.xx. An entry of “0000” or “9999” causes the integral gain to be “infinity“, effectively removing the integrator term from the PID equation. This accommodates applications which need proportional-only loops. The units of integral gain may be either seconds or minutes, as shown in the above dialog.

Rate (Derivative Gain) – Values can be entered in the range of 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 which 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 be set later in the DirectSOFT 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 is enabled, then: Error = SP–√PV. In any case, the size of the error and algebraic sign determine the next change of the control output for each PID calculation.

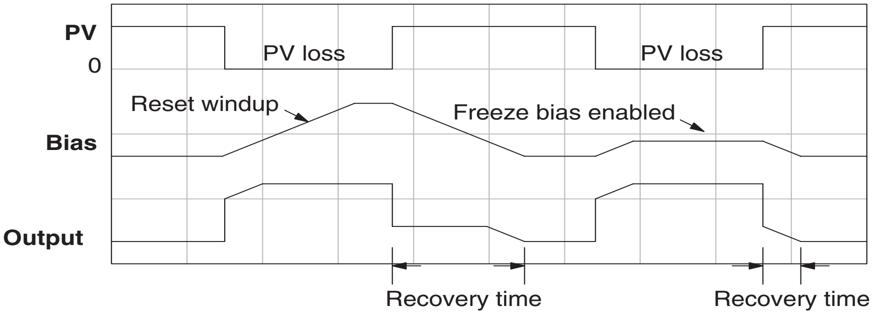
Error Squared – When selected, the squared error function simply squares the error term (but preserves the original algebraic sign), which is used in the calculation. This affects the Control Output by diminishing its response to smaller error values, but maintaining its response to larger errors. Some 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.

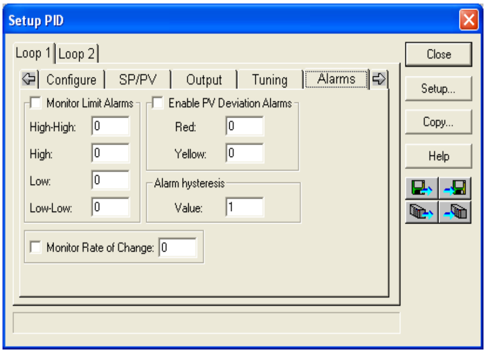
For most applications, the freeze bias feature will work with the loop as described above. It is suggested to enable this feature by selecting it in the dialog. Bit 10 of PID Mode 1 Setting (V+00) word can also be set in RLL.

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.*

Setup 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 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.



Setup PID

Loop 1 | Loop 2

Configure | SP/PV | Output | Tuning | Alarms

□ Monitor Limit Alarms
High-High: 0
High: 0
Low: 0
Low-Low: 0

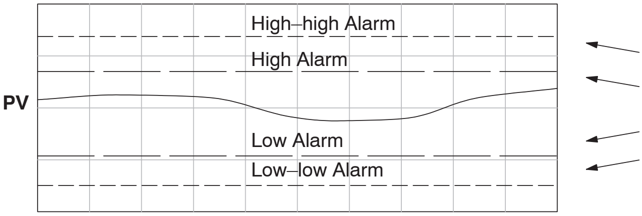
□ Enable PV Deviation Alarms
Red: 0
Yellow: 0

Alarm hysteresis
Value: 1

□ Monitor Rate of Change: 0

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.



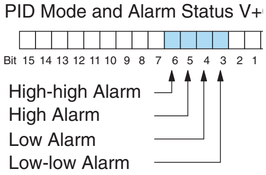
Loop Table

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 DirectSOFT PID View.*

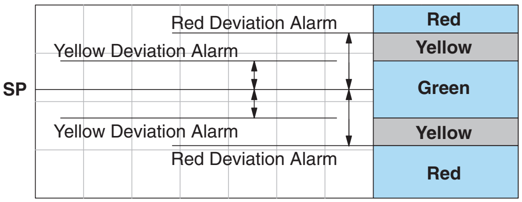
If the process remains out of control for some time, the PV will eventually cross one of the outer alarm thresholds, named High-high alarm and Low-low alarm. Their threshold values are programmed using the loop table registers listed above. A High-high or Low-low alarm indicates a serious condition exists, and needs the immediate attention of the operator.

The PV Absolute Value Alarms are reported in bits 3, 4, 5 and 6 of the PID Mode and Alarm Status word (V+06) 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 DirectSOFT.


PID Mode and Alarm Status V+
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
High-high Alarm
High Alarm
Low Alarm
Low-low Alarm

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.

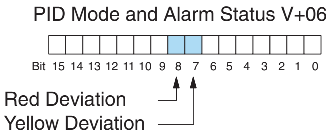


Loop Table

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 bits 7 and 8 of the PID Mode and Alarm Status word (V+06) 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 DirectSOFT.


PID Mode and Alarm Status V+06
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Red Deviation
Yellow Deviation

The PV Deviation Alarm can be independently enabled and disabled from the other PV alarms, using bit 13 of the PID Mode 1 Setting V+00 word.

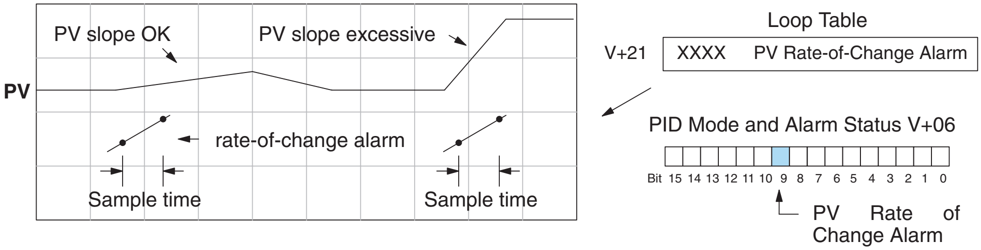
Remember the alarm hysteresis feature works in conjunction with both the deviation and absolute value alarms, and is discussed at the end of this section.

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, a 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 D4-454 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:

$$\text{Alarm Rate-of-Change} = \frac{\text{15 degrees}}{\text{1 minute}} \times \frac{\text{10 counts / degree}}{\text{30 loop samples / min.}} = \frac{150}{30} = \text{5 counts / sample period}$$

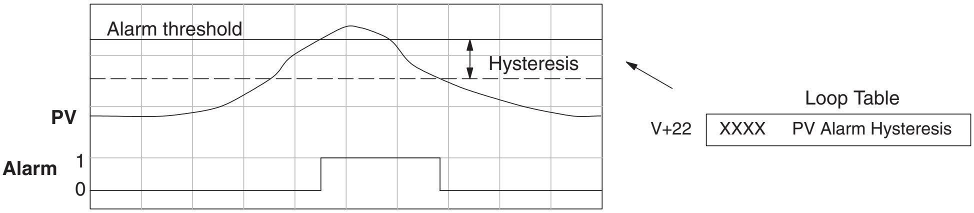
From the calculation result, you would program the value 5 in the loop table for the rate-of-change. The PV Rate-of-Change Alarm can be independently enabled and disabled from the other PV alarms, using bit 14 of the PID Mode 1 Setting V+00 word.

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.

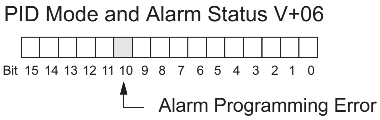
The PV Alarm Hysteresis amount is programmable from 1 to 200 (binary/decimal). When using the PV Deviation Alarm, the programmed hysteresis amount must be less than the programmed deviation amount. The figure below shows how the hysteresis is applied when the PV value goes past a threshold and descends 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

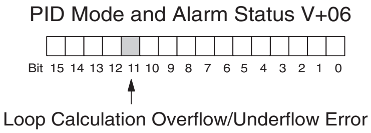
The PV Alarm threshold values must have certain mathematical relationships to be valid. The requirements are listed below. If not met, the Alarm Programming Error bit will be set, as indicated to the right.


PID Mode and Alarm Status V+06
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Alarm Programming Error

- 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.


PID Mode and Alarm Status V+06
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Loop Calculation Overflow/Underflow Error

NOTE: *Overflow/underflow can be alarmed in PID View. The optional C-more operator interface panel (see the automationdirect.com website) can also be setup 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.

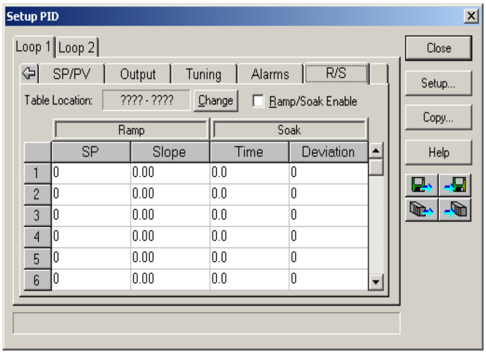


Table Location: ???? - ????

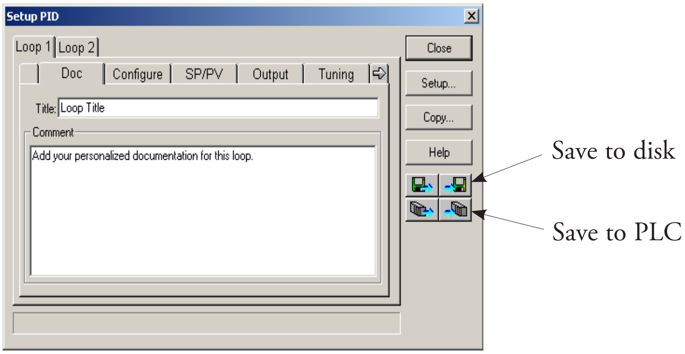
☐ Ramp/Soak Enable

	Ramp		Soak	
	SP	Slope	Time	Deviation
1	0	0.00	0.0	0
2	0	0.00	0.0	0
3	0	0.00	0.0	0
4	0	0.00	0.0	0
5	0	0.00	0.0	0
6	0	0.00	0.0	0

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.



Title: Loop Title

Comment

Add your personalized documentation for this loop.

***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 a 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 D4-454 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 D4-454 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.

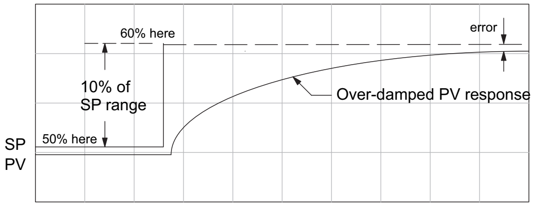
Manual Tuning Procedure

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.

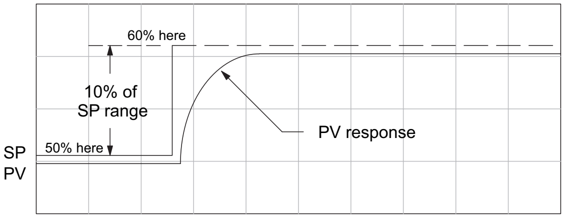
Monitor the values of SP, PV and CV with a loop trending instrument or use the PID View feature in DirectSOFT (see page 8-49).

***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.*

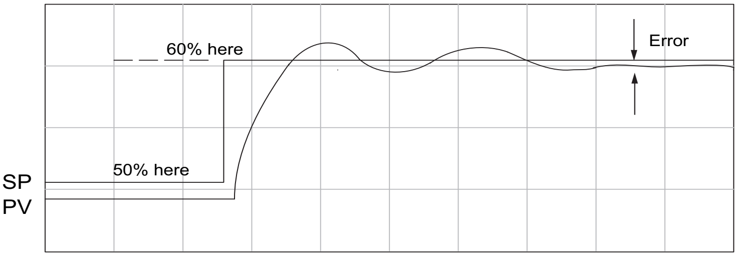
- Adjust the gains so the Proportional Gain = 0.5 or 1.0 (1.0 is a good value based on experience), Integral Gain = 9999 (this basically eliminates reset) and Derivative Gain = 0000. This disables the integrator and derivative terms, and provides some proportional gain.
- Check the bias value in the 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.



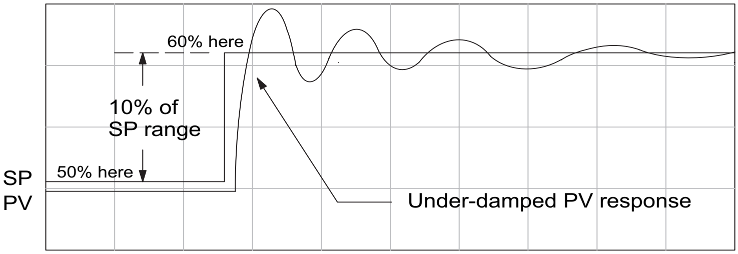
The response may take awhile, but you will see that there isn't any oscillation. This response is not desirable since it takes a long time to correct the error; also, there is a difference between the SP and the PV.



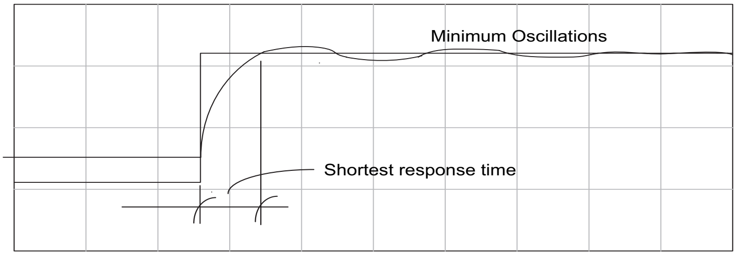
- Increase the Proportional gain, for example to 2.0. The control output will be greater and the response time will be quicker. The trend should resemble the figure below.
- Increase the Proportional gain in small increments, such as 4, 6, 7, etc., until the control output response begins to oscillate. 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 foregone 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; 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 (Gu).
4. Measure the period of cycling in minutes. This is the ultimate period (Pu).
5. Calculate the controller adjustments as follows:

P only: G = Gu/2

P & I : G = Gu/2.2
Ti = 1.2/Pu (repeats/minute)

P-I-D: G = Gu/1.6
Ti = 2.0/Pu (repeats/minute)
Td = Pu/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 = Gu/5.0
Ti = 3/Pu (repeats/minute)
Td = Pu/2 (minutes)

For some overshoot, but better response to disturbances:

G = Gu/3

Ti = 3/Pu (repeats/minute)

Td = Pu/3 (minutes)

Auto Tuning Procedure

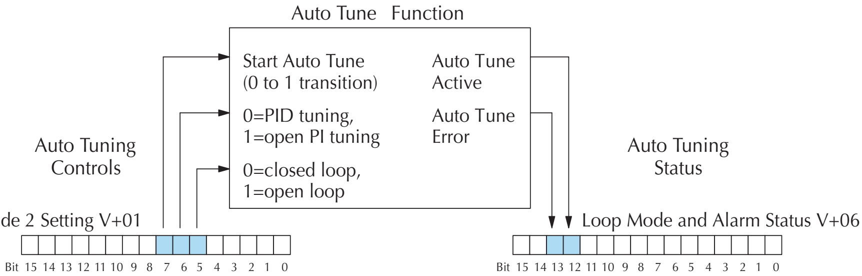
The auto tuning feature for the D4-454 loop controller will only run once each time it is enabled in the PID table. Therefore, auto tuning does not run continuously during operation (this would be adaptive control). Whenever there is a substantial change in loop dynamics, such as mass of process, size 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 D4-454 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 DirectSOFT (see the DirectSOFT 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.

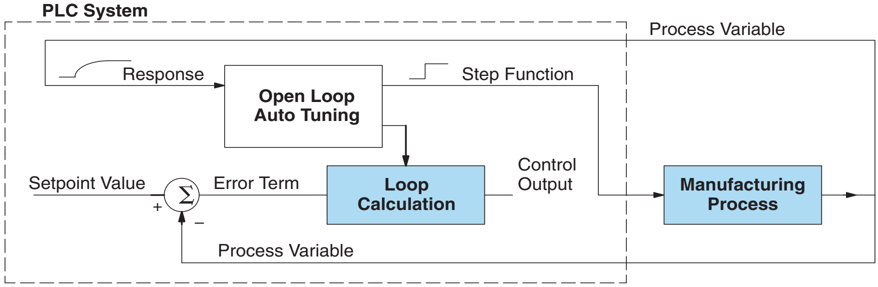
The loop controller offers both closed-loop and open-loop methods. The following sections describe how to use the auto tuning feature, and what occurs in open and closed-loop auto tuning.

The controls for the auto tuning function uses bits 5, 6, and 7 in the PID Mode 2 word V+01, as shown below. DirectSOFT will manipulate these bits automatically when you use the auto tune feature within DirectSOFT. 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).

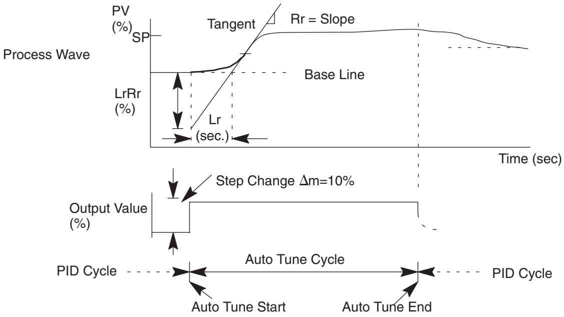


NOTE: *In theory, the SP value does not matter in this case, because the loop is not closed. However, the requirement of the firmware is that the SP value must be more than 5% of the PV range from the actual PV before starting the auto tune cycle (for the D4-454, 12 bit PV should be 205 counts or more 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.



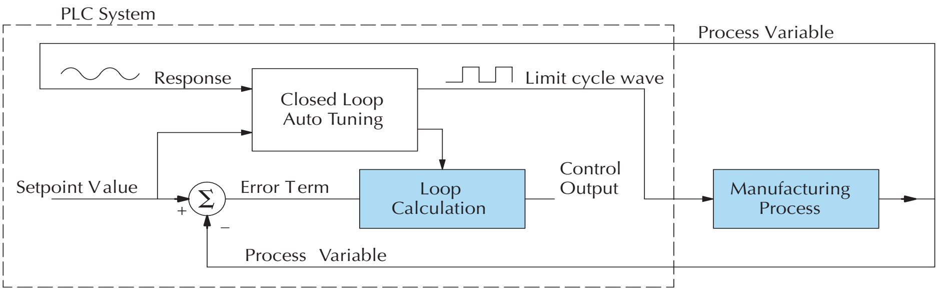
When the loop tuning observations are complete, the loop controller computes Rr (maximum slope in %/sec.) and Lr (dead time in sec). The auto tune function computes the gains according to the Zeigler-Nichols equations, shown below:

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 DirectSOFT 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 diagram below.

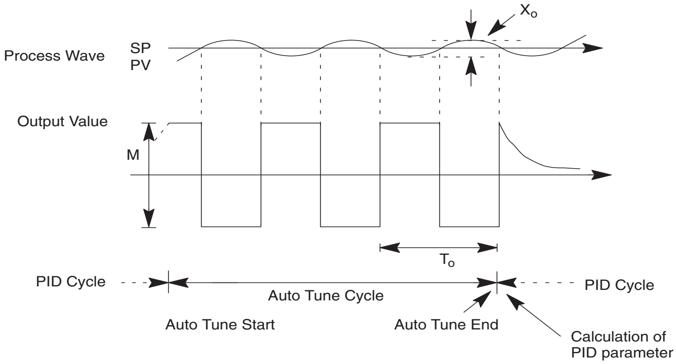


When auto tuning, the loop controller imposes a square wave on the output. Each transition of the output occurs when the PV value crosses over/under the SP value. Therefore, the frequency of the limit cycle is roughly proportional to the mass of the process. From the PV response, the auto tune function 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.

This example is direct–acting.

- Mmax = Output Value upper limit setting.
- Mmin = Output Value lower limit setting.



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:

$$\text{Kpc} = 4\text{M} / (\pi * \text{X0}) \qquad \text{Tpc} = 0$$

M = Amplitude of output

PID Tuning

P = 0.45*Kpc

I = 0.60*Tpc

D = 0.10*Tpc

Sample Rate = 0.014*Tpc

PI Tuning

P = 0.30*Kpc

I = 1.00*Tpc

D = 0

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, please verify the PV and SP values are within 5% of full scale difference, as required by the auto tune function.

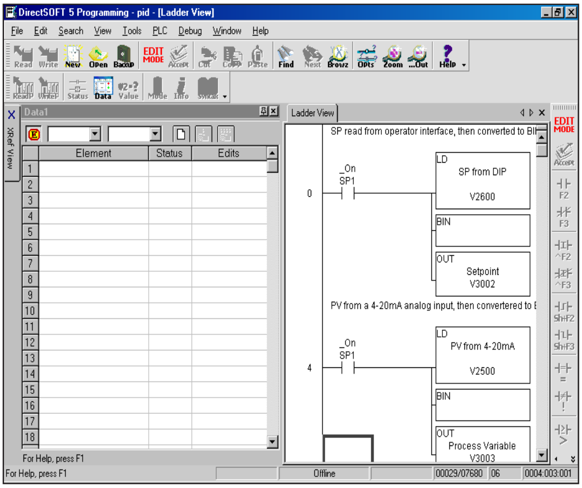
***NOTE:** If your PV fluctuates rapidly, you probably need to use the built-in analog filter (see page 8–55) or create a filter in ladder logic (see example on page 8–56).*

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.


DirectSOFT 5 Programming - pid - [Ladder View]
File Edit Search View Tools PLC Debug Window Help
Read Write New Open Backup EDIT MODE Accept Cut Copy Paste Find Next Browz Opts Zoom ...Out Help
ReadP WriteP Status Data Value Mode Info Syncall
XRef View
Data1
Element Status Edits
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
For Help, press F1
Ladder View
EDIT MODE
Accept
F2 F3 ^F2 ^F3 ShF2 ShF3
For Help, press F1
Offline
00029/07680 06
0004:003:001

SP read from operator interface, then converted to BI

0 _On SP1 — LD SP from DIP V2600; BIN; OUT Setpoint V3002

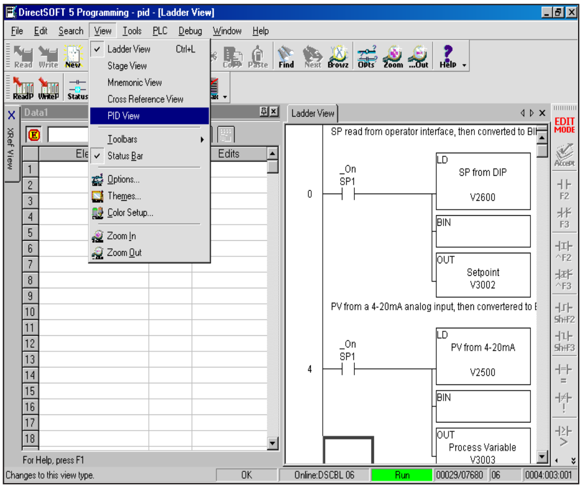
PV from a 4-20mA analog input, then convertered to E

4 _On SP1 — LD PV from 4-20mA V2500; BIN; OUT Process Variable V3003

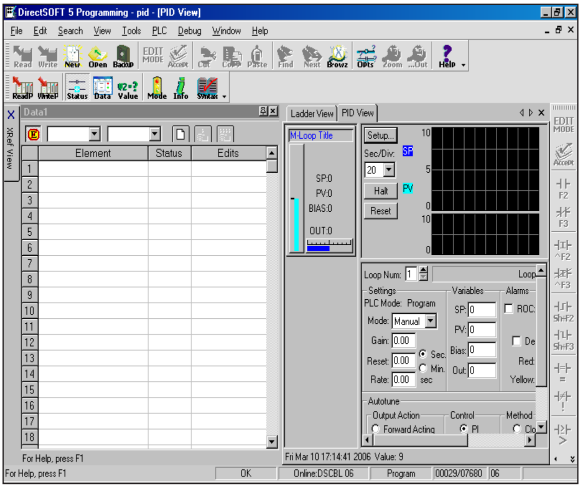
The Data View window can be used just as it is shown above for troubleshooting your PID logic, and it can be most useful when tuning the PID loop.

Open PID View

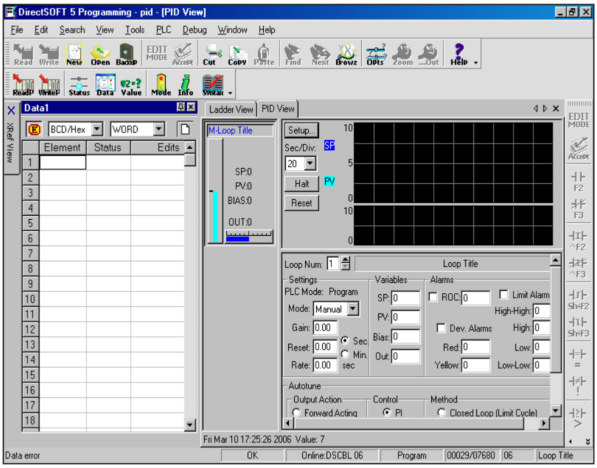
The PID View can only be opened after a loop has been setup in your ladder program. PID View is opened by selecting it from the View submenu on the Menu bar, View > PID View. The PID View can also be opened by clicking on the PID View button from the PLC Setup toolbar if it is in view.



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.


DirectSOFT 5 Programming - pid - [PID View]
File Edit Search View Tools PLC Debug Window Help
Read Write New Open Backup EDIT MODE Accept Cut Copy Paste Find Next Browz Opts Zoom ...Out Help
ReadP WriteP Status Data Value Mode Info Syntax
Data1
BCD/Hex WORD
Element Status Edits
Ladder View PID View
M-Loop Title
Setup... Sec/Div: 20 Halt Reset SP PV
SP:0 PV:0 BIAS:0 OUT:0
10 5 0 10 0
EDIT MODE Accept F2 F3 ^F2 ^F3 ShF2 ShF3

Loop Num: 1 Loop Title

Settings: PLC Mode: Program; Mode: Manual; Gain: 0.00; Reset: 0.00 ● Sec. ○ Min.; Rate: 0.00 sec

Variables: SP: 0; PV: 0; Bias: 0; Out: 0

Alarms: □ ROC: 0; □ Limit Alarm: High-High: 0, High: 0, Low: 0, Low-Low: 0; □ Dev. Alarms: Red: 0, Yellow: 0

Autotune: Output Action: ○ Forward Acting; Control: ● PI; Method: ○ Closed Loop (Limit Cycle)

Fri Mar 10 17:25:26 2006 Value: 7

Data error | OK | Online:DSCBL 06 | Program | 00029/07680 | 06 | Loop Title

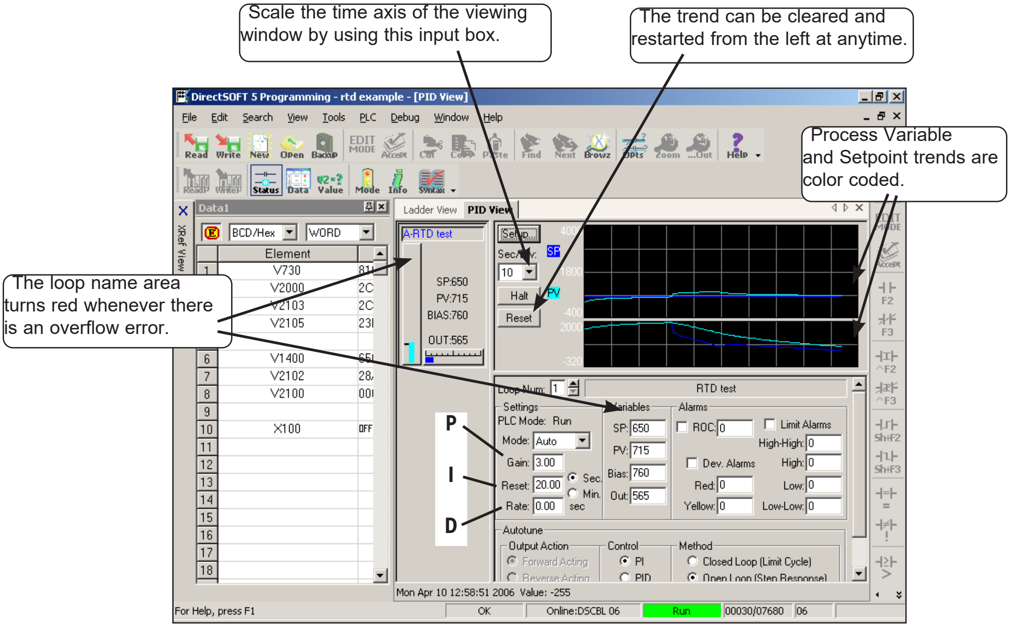
The diagram below illustrates how to use the views to see the current SP, PV and Output values, along with the other PID addresses. Refer to the Loop Table Definitions page 8-21 for details of each word in the table. This is also a good data type reference for each word in the table.

Scale the time axis of the viewing window by using this input box.

The trend can be cleared and restarted from the left at anytime.

Process Variable and Setpoint trends are color coded.

The loop name area turns red whenever there is an overflow error.


DirectSOFT 5 Programming - rtd example - [PID View]
File Edit Search View Tools PLC Debug Window Help
Read Write New Open Backup EDIT MODE Accept Cut Copy Paste Find Next Browz Opts Zoom ...Out Help
ReadP WriteP Status Data Value Mode Info Syntax
XRef View
Data1
BCD/Hex WORD
Ladder View PID View
A-RTD test
Setup... Sec/Div: 10 Halt Reset SP PV
400 1800 -400 2000 -320
P
I
D
EDIT MODE Accept F2 F3 ^F2 ^F3 ShF2 ShF3

	Element	
1	V730	81
2	V2000	2C
3	V2103	2C
4	V2105	23I
6	V1400	65
7	V2102	28
8	V2100	00
9		
10	X100	OFF
11		
12		
13		
14		
15		
16		
17		
18		

SP:650 PV:715 BIAS:760 OUT:565

Loop Num: 1 RTD test

Settings: PLC Mode: Run; Mode: Auto; Gain: 3.00; Reset: 20.00 ● Sec. ○ Min.; Rate: 0.00 sec

Variables: SP: 650; PV: 715; Bias: 760; Out: 565

Alarms: □ ROC: 0; □ Limit Alarms: High-High: 0, High: 0, Low: 0, Low-Low: 0; □ Dev. Alarms: Red: 0, Yellow: 0

Autotune: Output Action: ● Forward Acting, ○ Reverse Acting; Control: ● PI, ○ PID; Method: ○ Closed Loop (Limit Cycle), ● Open Loop (Step Response)

Mon Apr 10 12:58:51 2006 Value: -255

For Help, press F1 | OK | Online:DSCBL 06 | Run | 00030/07680 | 06

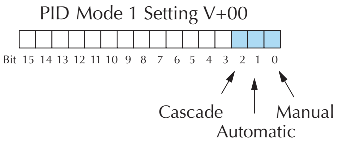
When both windows are positioned in this manner, you are able to see where the PID values have been set and see the process that it is controlling.

Using the Special PID Features

It’s a good idea to understand the special features of the D4-454 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 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).

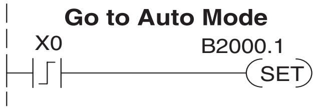


The normal state of these mode request bits is “000.”

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:

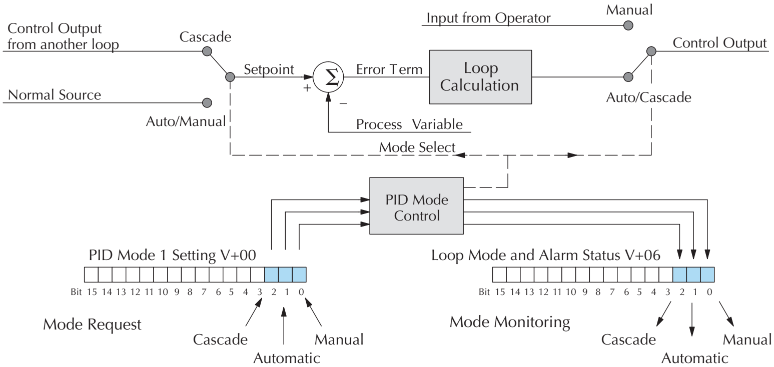
- DirectSOFT’s PID View – this is the easiest method. Use the pull-down menu, or click on one of the radio buttons if using older DirectSOFT versions, and the appropriate bit will get set.
- Ladder program– ladder logic can request any loop mode when the PLC is in Run Mode. This will be necessary after application startup if mode changes are part of the application.

Use the program 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 above to set the mode bit.

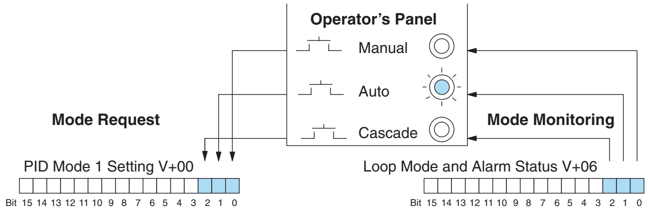
Since mode changes can only be requested, the PID loop controller will decide when to permit mode changes and provide the loop mode status. It reports the current mode on bits 0, 1, and 2 of the Loop Mode/Alarm Status word, location V+06 in the loop table. The parallel request/monitoring functions are 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.


Operator's Panel
Manual
Auto
Cascade
Mode Request
Mode Monitoring
PID Mode 1 Setting V+00
Loop Mode and Alarm Status V+06
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

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 16 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:

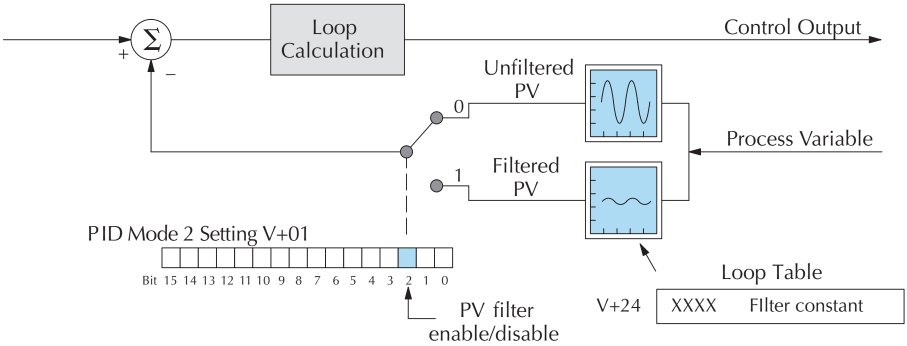
- A loop which develops an error condition automatically goes to Manual.
- If the minor loop of a cascaded pair of loops leaves Cascade Mode for any reason, its major loop automatically goes to Manual Mode.

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 D4-454's built-in filter. The second method achieves a similar result using ladder logic.

The D4-454 Built-in Analog Filter

The D4-454 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.
- DirectSOFT 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.

The algorithm which the built-in filter follows is:

yi = k (xi – yi–1) + yi–1

yi is the current output of the filter

xi is the current input to the filter

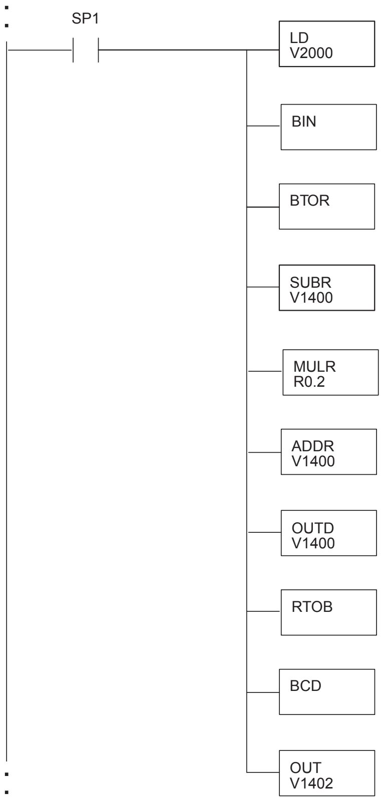
yi–1 is the previous output of the filter

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.



Loads the analog signal, which is a BCD value and has been loaded from V-memory location V2000, into the accumulator. Contact SP1 is always on.

Converts the BCD value in the accumulator to binary. This instruction is not needed if the analog value is originally brought in as a binary number.

Converts the binary value in the accumulator to a real number.

Subtracts the real number stored in location V1400 from the real number in the accumulator, and stores the result in the accumulator. V1400 is the designated workspace in this example.

Multiplies the real number in the accumulator by 0.2 (the filter factor), and stores the result in the accumulator. This is the filtered value.

Adds the real number stored in location V1400 to the real number filtered value in the accumulator, and stores the result in the accumulator.

Copies the value in the accumulator to location V1400.

Converts the real number in the accumulator to a binary value, and stores the result in the accumulator.

Converts the binary value in the accumulator to a BCD number. Note: the BCD instruction is not needed for PID loop PV (loop PV is a binary number).

Loads the BCD number filtered value from the accumulator into location V1402 to use in your application or PID loop.

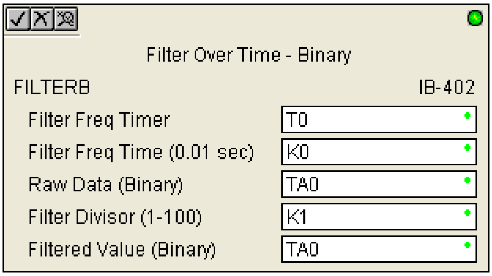
Use the DirectSOFT Filter Intelligent Box Instruction

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.



Filter Over Time - Binary

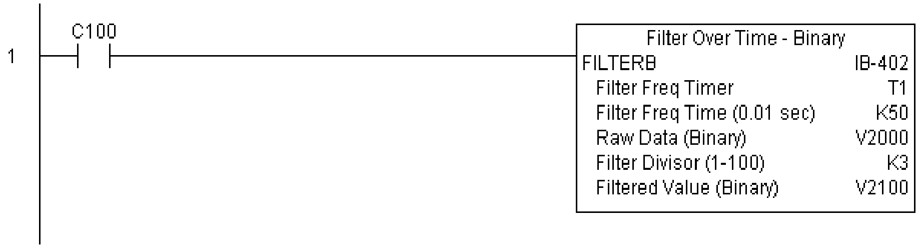
FILTERB	IB-402
Filter Freq Timer	T0
Filter Freq Time (0.01 sec)	K0
Raw Data (Binary)	TA0
Filter Divisor (1-100)	K1
Filtered Value (Binary)	TA0

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, use ladder to move the Filtered Value result to the PID PV location (V+03).

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.



Filter Over Time - Binary

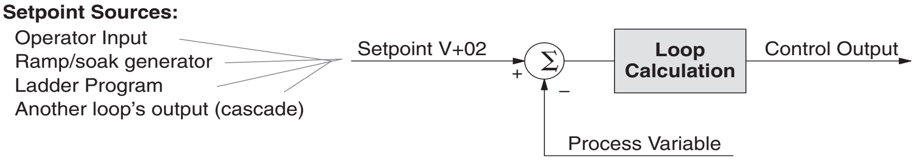
FILTERB	IB-402
Filter Freq Timer	T1
Filter Freq Time (0.01 sec)	K50
Raw Data (Binary)	V2000
Filter Divisor (1-100)	K3
Filtered Value (Binary)	V2100

NOTE: *For information on the Filter Over Time-Binary Ibox IB-402, see the DL405-IBOX-S manual.*

Ramp/Soak Generator

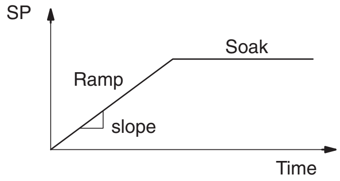
Introduction

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.



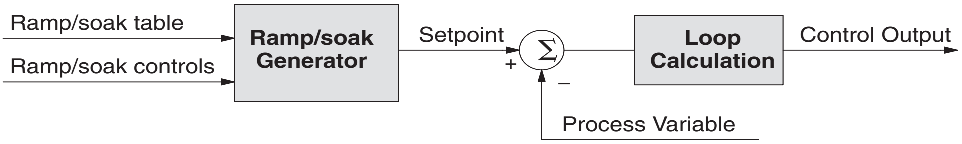
If the SP for your process rarely changes or can tolerate step changes, you probably will not need to use the ramp/soak generator. However, some processes require precisely-controlled SP value changes. The ramp/soak generator can greatly reduce the amount of programming required for these applications.

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.

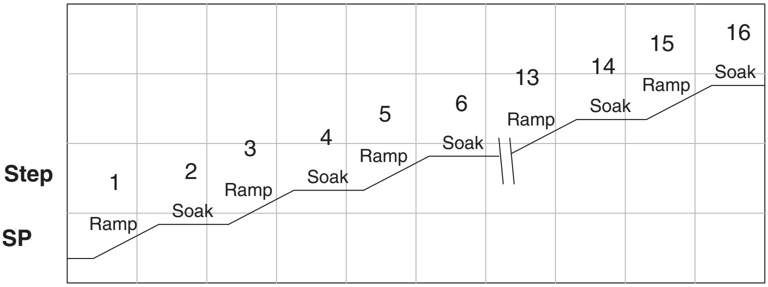
It is instructive to view the ramp/soak generator as a dedicated function to generate SP values, as shown below. It has two categories of inputs which determine the SP values generated. The ramp/soak table must be programmed in advance, containing the values that will define the ramp/soak profile. The 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.

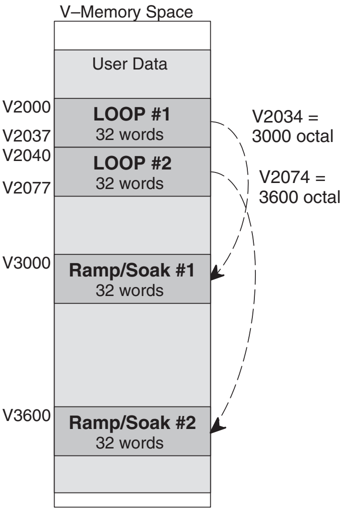
The following figure shows an SP profile consisting of ramp/soak segment pairs. The segments are individually numbered as steps from 1 to 16. The slope of each of the ramp segments may be either increasing or decreasing. The ramp/soak generator automatically knows whether to increase or decrease the SP based on the relative values of a ramp's end points. These values come from the ramp/soak table.



Ramp/Soak Table

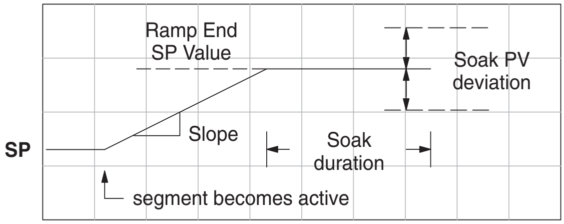
The parameters which define the ramp/soak profile for a loop are in a ramp/soak table. Each loop may have its own ramp/soak table, but it is optional. Recall the Loop Parameter table consists of a 32-word block of memory for each loop, and together they occupy one contiguous memory area. However, the 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 DirectSOFT, 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 R/S steps. Use all zeros in the table for unused steps. The R/S generator ends the profile when it finds ramp slope = 0.

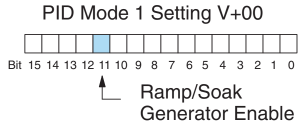
Ramp/Soak Table Flags

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	–	0 –> 1 Start
1	Hold Ramp / Soak Profile	Write	–	0 –> 1 Hold
2	Resume Ramp / Soak Profile	Write	–	0 –> 1 Resume
3	Jog Ramp / Soak Profile	Write	–	0 –> 1 Jog
4	Ramp / Soak Profile Complete	Read	–	Complete
5	PV Input Ramp / Soak Deviation	Read	Off	On
6	Ramp / Soak Profile in Hold	Read	Off	On
7	Reserved	Read	Off	On
8–15	Current Step in R/S Profile	Read	Decode as byte (hex)	

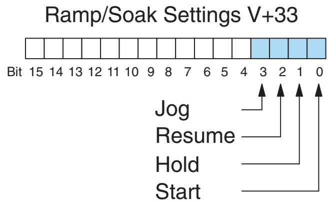
Ramp/Soak Generator Enable

The main enable control to permit ramp/soak generation of the SP value is accomplished with bit 11 in the PID Mode 1 Setting 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.



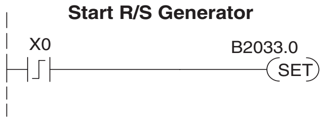
Ramp/Soak Controls

The four main controls for the ramp/soak generator are in bits 0 to 3 of the Ramp/Soak Setting V + 33 word in the loop parameter table. DirectSOFT 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.

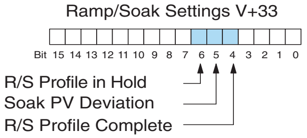


The normal state for the ramp/soak control bits is all zeros. Ladder logic must set only one control bit at a time.

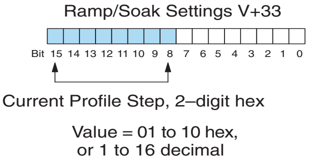
- Start – a 0 to 1 transition will start the ramp soak profile. The CPU must be in Run Mode, and the loop can be in Manual or Auto Mode. If the profile is not interrupted by a Hold or Jog command, it finishes normally.
- Hold – a 0 to 1 transition will stop the ramp/soak profile in its current state, and the SP value will be frozen.
- Resume – a 0 to 1 transition will cause the ramp/soak generator to resume operation if it is in the hold state. The SP values will resume from their previous value.
- 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.

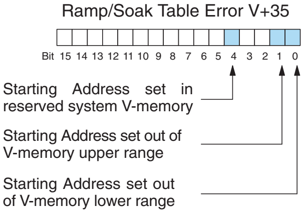


- R/S Profile Complete =1 when the last programmed step is done.
- Soak PV Deviation =1 when the error (SP–PV) exceeds the specified deviation in the R/S table.
- R/S Profile in Hold =1 when the profile was active but is now in hold.
- The number of the current step is available in the upper 8 bits of 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 Settings V+33
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Current Profile Step, 2–digit hex
Value = 01 to 10 hex,
or 1 to 16 decimal

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 DirectSOFT to configure the ramp/soak table. It automatically range checks the addresses for you.


Ramp/Soak Table Error V+35
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Starting Address set in reserved system V-memory
Starting Address set out of V-memory upper range
Starting Address set out of V-memory lower range

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 DirectSOFT'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.

Setup the Profile in PID Setup

The first step is to use Setup PID in DirectSOFT 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.

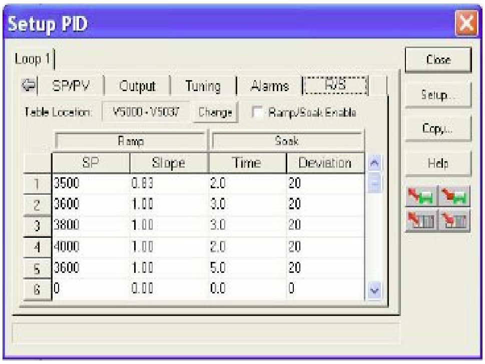
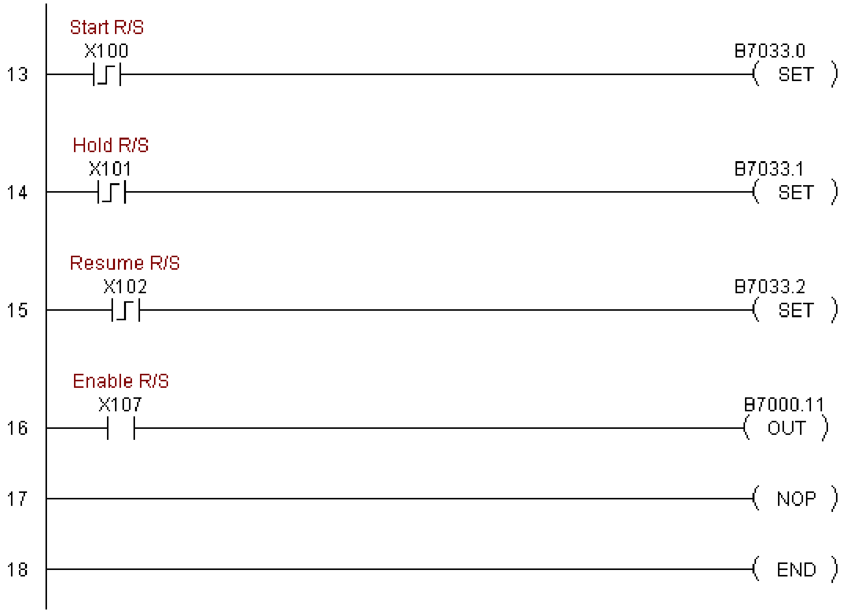


Table Location: V5000 - V5037 Change
□ Ramp/Soak Enable

	Ramp		Soak	
	SP	Slope	Time	Deviation
1	3500	0.83	2.0	20
2	3600	1.00	3.0	20
3	3800	1.00	3.0	20
4	4000	1.00	2.0	20
5	3600	1.00	5.0	20
6	0	0.00	0.0	0

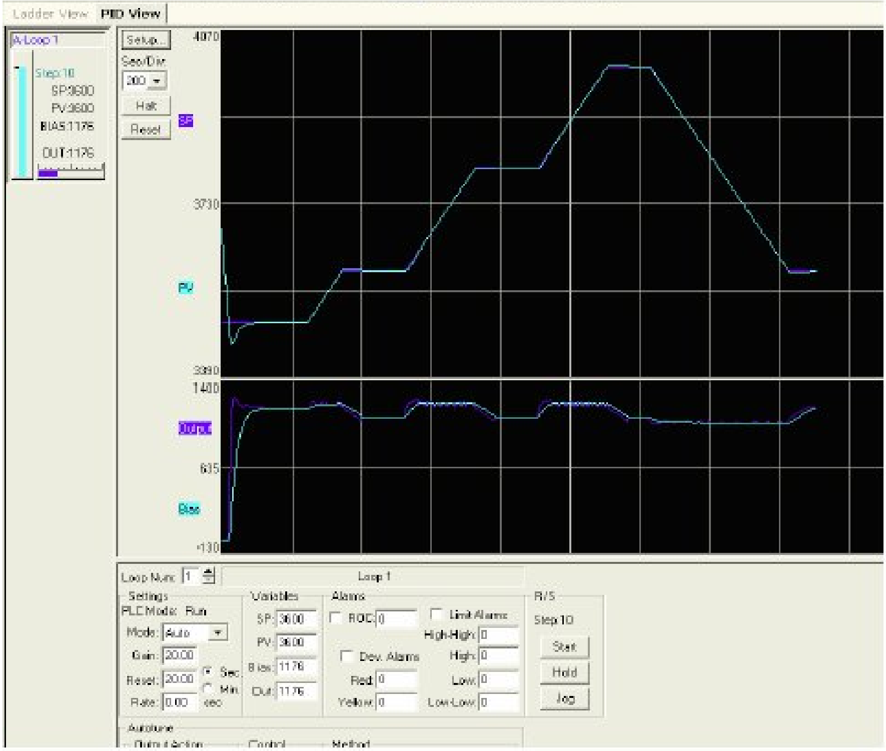
Program the Ramp/Soak Control in Relay Ladder

Refer to the Ramp/Soak Flag Bit Description table on page 8-63 when adding the control rungs to your program similar to the ladder rungs below.



Test the Profile

After the Ramp/Soak program has been developed in RLL, test the program. Check your profile by using PID View. If there are any changes to be made in the profile, they are made in the PID Setup R/S profile. Make the changes in Program mode then start the Ramp/Soak process again.



A-Loop 1
Step:10
SP:3600
PV:3600
BIAS:1176
OUT:1176

Sec/Div: 200

Loop Num: 1
Loop 1

Settings
PLC Mode: Run
Mode: Auto
Gain: 20.00
Reset: 20.00 ● Sec. ○ Min.
Rate: 0.00 sec

Variables
SP: 3600
PV: 3600
Bias: 1176
Out: 1176

Alarms
□ ROC: 0
□ Dev. Alarms
Red: 0
Yellow: 0
□ Limit Alarms
High-High: 0
High: 0
Low: 0
Low-Low: 0

R/S
Step 10

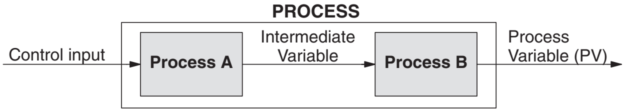
Cascade Control

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 D4-454 also provides Cascaded Mode.

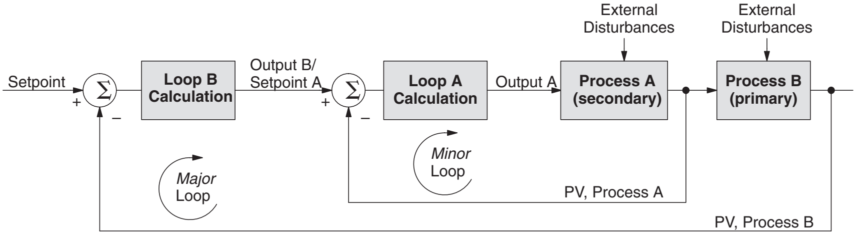
NOTE: *Using cascaded loops is an advanced process control technique; therefore, we recommend their use only for experienced process control engineers.*

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.

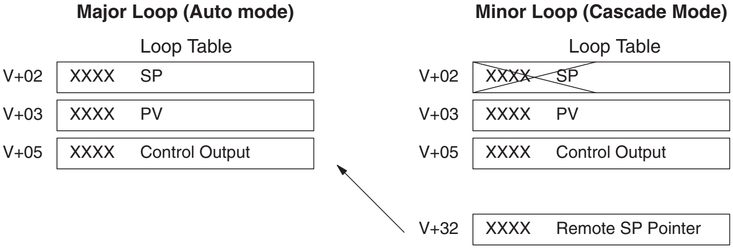
Cascaded Loops in the D4-454 CPU

In the use 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 outer-most (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 outer-most (major) loop will be in Auto Mode.*

You can cascade together as many loops as necessary on the D4-454, 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 loop.

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 its local SP register (V+02), and read the major loop's control output as its SP instead.



Major Loop (Auto mode)

Loop Table

V+02	XXXX	SP
V+03	XXXX	PV
V+05	XXXX	Control Output

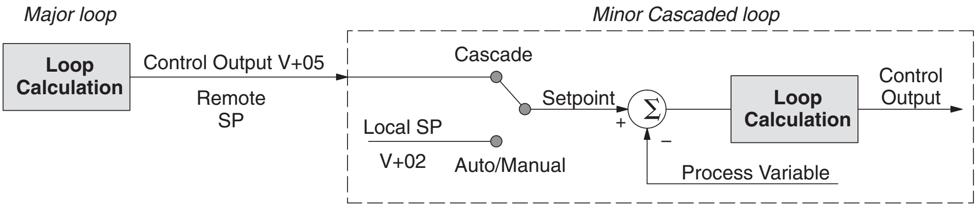
Minor Loop (Cascade Mode)

Loop Table

V+02	~~XXXX~~	~~SP~~
V+03	XXXX	PV
V+05	XXXX	Control Output
V+32	XXXX	Remote SP Pointer

When using DirectSOFT's PID View to watch the SP value of the minor loop, DirectSOFT 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.

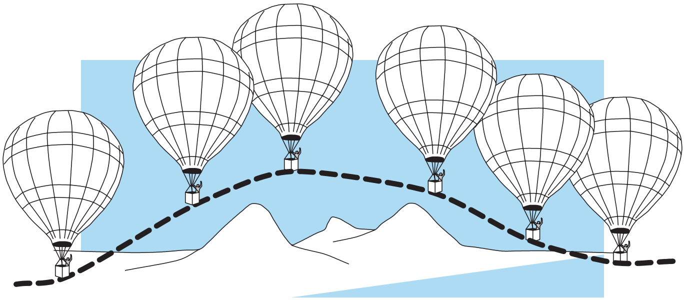
1. If you are not using auto tuning, then find the loop sample rate for the minor loop, using the method discussed earlier in this chapter. Then set the sample rate of the major loop slower than the minor loop by a factor of 10. Use this as a starting point.
2. Tune the minor loop first. Leave the major 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 in 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

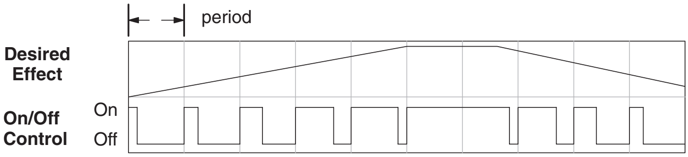
The PID loop controller in the D4-454 CPU generates a smooth control output signal across a numerical range. The control output value is suitable to drive an analog output module, which connects to the process. In the process control field, this is called continuous control, because the output is on (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.



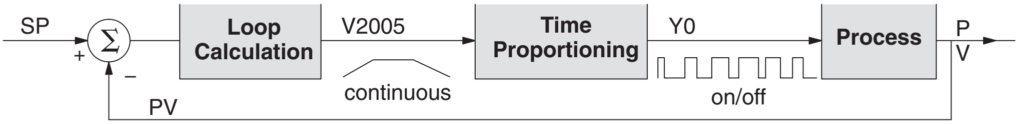
Time-proportioning control approximates continuous control by virtue of its duty-cycle the ratio of ON time to OFF time. The following figure shows an example of how duty-cycle approximates a continuous level when it is averaged by a large process mass.



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.

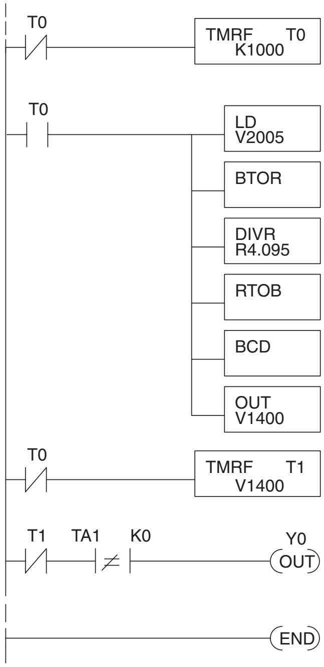


The example program uses two timers to generate On/Off control. It makes the following assumptions, which you can alter to fit your application:

- 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 or 0 – 4095).
- 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 “matchs” 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.

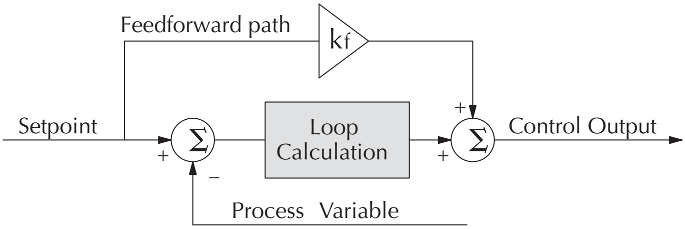
The second fast timer also counts in increments of .01 seconds, so its range is variable from 0 to a maximum of 1000 ticks, or 10 seconds. This timer’s output, T1, turns off the output coil, Y0, when the preset is reached.

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 D4-454. 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.

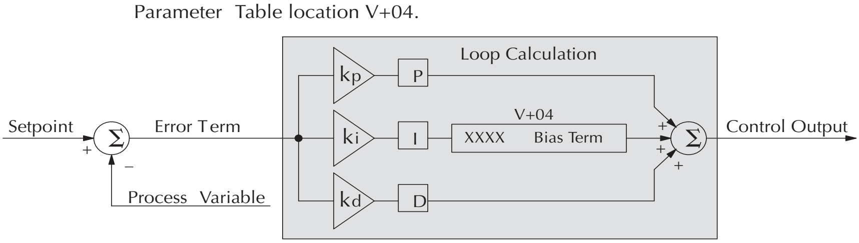


In the previous section on the bias term, we said that "the bias term value establishes a working region or operating point for the control output. When the error fluctuates around its zero point, the output fluctuates around the bias value." Now, when there is a change in setpoint, an error is generated 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 D4-454 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.

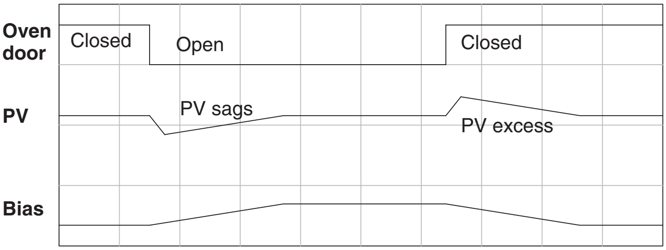


To change the bias (operating point), ladder logic only has to write the desired value to V+04. The PID loop calculation first reads the bias value from V+04 and modifies the value based on the current integrator calculation. Then it writes the result back to location V+04. This arrangement creates a 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).

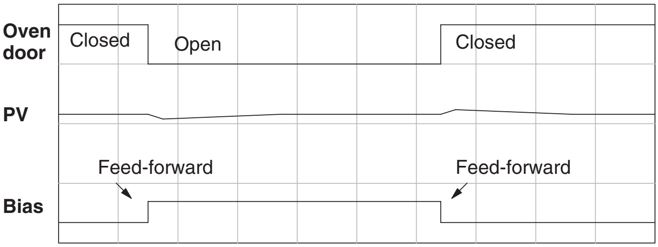
***NOTE:** When writing the bias term, one must be careful to design ladder logic to write the value only once, at the moment when the new bias operating point is to occur. If ladder logic writes the bias value on every scan, the loop's integrator is effectively disabled.*

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.

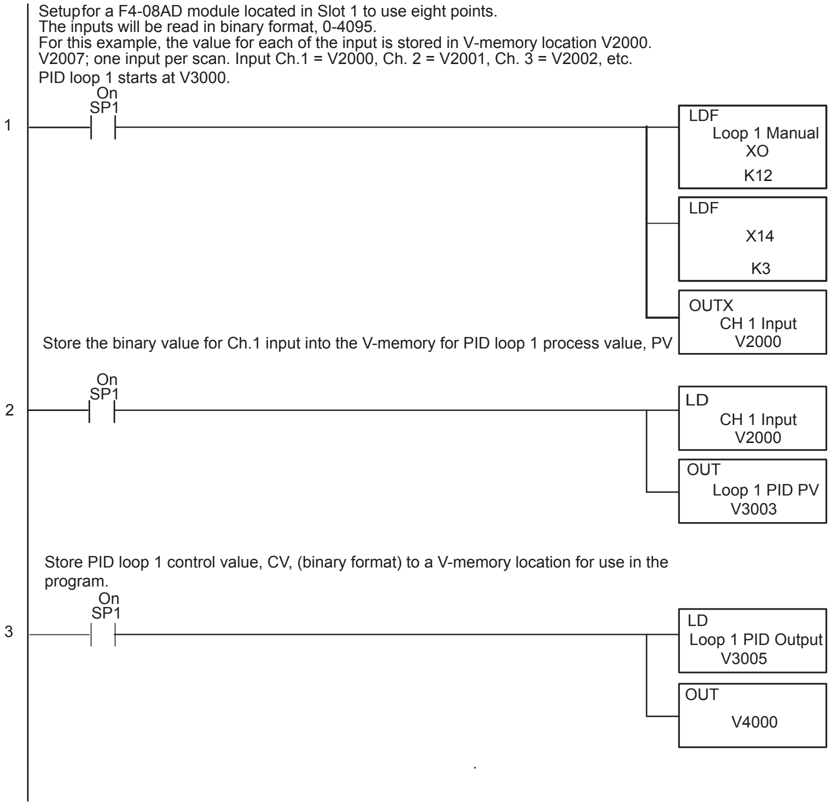
PID Example Program

Program Setup for the PID Loop

After the PID loop(s) has been setup with DirectSOFT, you will need to edit your RLL program to include the rungs needed to setup the analog I/O module to be used by the PID loop(s).

The example program shows how an analog input module, F4-08AD is used to setup a PID loop. This example assumes that the PID table for loop 1 has a beginning address of V3000.

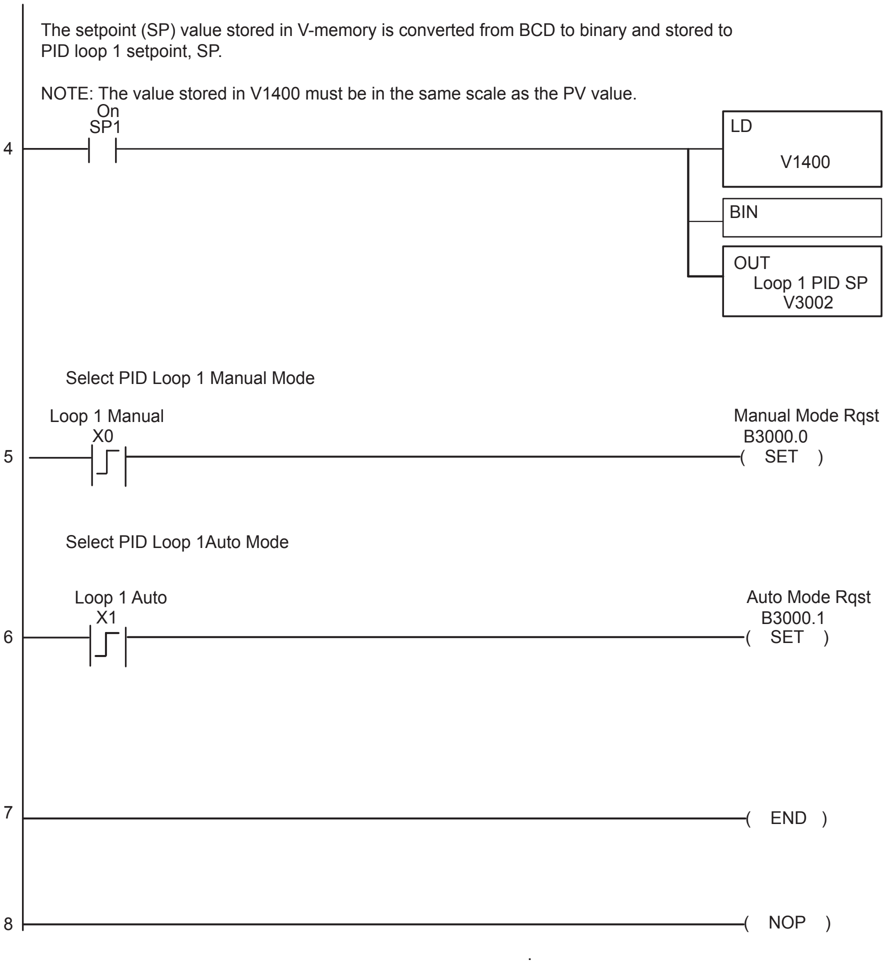
All the analog I/O modules used with the D4-454 are setup in a similar manner. Refer to the DL405 Analog I/O Manual for the setup information for the module that you will be using.



Setupfor a F4-08AD module located in Slot 1 to use eight points.
The inputs will be read in binary format, 0-4095.
For this example, the value for each of the input is stored in V-memory location V2000.
V2007; one input per scan. Input Ch.1 = V2000, Ch. 2 = V2001, Ch. 3 = V2002, etc.
PID loop 1 starts at V3000.

Store the binary value for Ch.1 input into the V-memory for PID loop 1 process value, PV

Store PID loop 1 control value, CV, (binary format) to a V-memory location for use in the program.



The setpoint (SP) value stored in V-memory is converted from BCD to binary and stored to PID loop 1 setpoint, SP.

NOTE: The value stored in V1400 must be in the same scale as the PV value.

Select PID Loop 1 Manual Mode

Select PID Loop 1Auto Mode

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 later 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 binary to BCD before being stored to the analog output.

By following the steps outlined in this chapter, you should be able to setup workable PID control loops. The DirectSOFT 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, 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 Powerpoint 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 DirectSOFT, 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 DirectSOFT 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 DirectSOFT converts them for you. The values in the table range from 0 to FFF, for 12-bit unipolar format. Your ladder program must convert constant values from their BCD format (when entered as Kxxxx) to binary with the BIN instruction or you must enter them in the constant field (Kxxxx) as the hex equivalent of the decimal value.

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 Process: Control 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 D4-454'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

Chapter 9

In This Chapter...

Hardware System Maintenance

Standard Maintenance

No regular or preventative maintenance is required for this product (there are no internal batteries); however, a routine maintenance check (about every one or two months) of your PLC and control system is good practice, and should include the following items:

- Air Temperature – Monitor the air temperature in the control cabinet, so the operating temperature range of any component is not exceeded.
- Air Filter – If the control cabinet has an air filter, clean or replace it periodically as required.
- Fuses or breakers – Verify that all fuses and breakers are intact.
- Cleaning the Unit – Check that all air vents are clear. If the exterior case needs cleaning, disconnect the input power, and carefully wipe the case using a damp cloth. Do not let water enter the case through the air vents and do not use strong detergents because this may discolor the case.

Battery Replacement

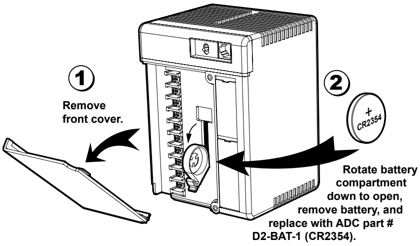
The CPU battery is used to retain program V-memory and the system parameters. The life expectancy of this battery is 5 years.

NOTE: *Before installing or replacing your CPU battery, back-up your V-memory and system parameters. You can do this by using DirectSOFT programming software to save the program, V-memory, and system parameters to a personal computer.(File > Save Project to disk.)*

To install the D2–BAT–1 (#CR2354) CPU battery in the D4-454 CPUs.

- Press the retaining clip on the battery door down and swing the battery door open.
- Remove old battery and insert the new battery into the coin–type slot with the larger (+) side outwards
- Close the battery door making sure that it locks securely in place.
- 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.


1
Remove front cover.
2
+
CR2354
Rotate battery compartment down to open, remove battery, and replace with ADC part # D2-BAT-1 (CR2354).

Diagnostics

Your D4-454 PLC performs many predefined diagnostic routines with every CPU scan. The diagnostics can detect various errors or failures in the PLC. The two primary error classes are fatal and non-fatal.

Fatal Errors

Fatal errors are errors which may cause the system to function improperly, perhaps introducing a safety problem. The CPU will automatically switch to Program Mode if it is in Run 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 allow you to transition to Run Mode until the error has been corrected.

Some examples of fatal errors are:

- Power supply failure on the CPU base
- Parity error or CPU malfunction
- Particular programming errors

Non-fatal Errors

Non-fatal errors are errors that need your attention, but should not cause improper operation. They do not cause or prevent any mode transitions of the CPU. The application program can use special relay contacts to detect non-fatal errors, and even take the system to an orderly shutdown or switch the CPU to Program Mode if desired. An example of a non-fatal error is:

- Particular programming errors - The programming devices will notify you of an error if one occurs while online.
- DirectSOFT programming software provides the error number and an error message.
- Backup battery voltage low
- All I/O module errors

See the Error Codes Appendix for a complete list of error messages in order by error number. Many error messages point to supplemental V-memory locations which contain related information. Special relays (SP contacts) also provide error indications (refer to the Special Relays Appendix).

V-memory Error Code Locations

The following table names the specific memory locations that correspond to certain types of error messages.

Error Class	Error Category	Diagnostic V-memory
Minor	Battery Voltage	V7746
System	10ms calendar timer	V7747
User-Defined	Error code used with FAULT instruction	V7751
I/O Configuration	Current module ID Code	V7752
		V7753
	Base number/Slot number	V7754
System Error	Fatal Error code	V7755
	Major Error code	V7756
	Minor Error code	V7757
Module	Base Number/Slot number	V7760
Module	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

The special relay table also includes status indicators which can indicate errors. For a more detailed description of each of these special relays refer to the Special Relays Appendix.

CPU Status Relays	
SP11	Forced Run mode
SP12	Terminal Run mode
SP13	Test Run mode
SP14	Break Relay 1
SP15	Test Stop mode
SP16	Terminal Program mode
SP17	Forced Sop mode
SP21	Break relay 2
SP22	Interrupt enabled
SP25	CPU battery disabled
SP30	Dip switch 1 status
SP31	Dip switch 2 status
SP32	Dip switch 3 status
SP33	Dip switch 4 status

Startup and Real-time Relays	
SP0	First scan
SP1	Always ON
SP3	1 minute
SP4	1 second
SP5	100 millisecond
SP6	50 millisecond
SP7	Alternate scan

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	Communication error
SP56	Table instruction overrun

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	
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 4
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 tables list these codes.

Code (Hex)	Component Type
01	CPU
02	Expansion Unit
03	I/O Base
11	DCM, All CoProcessor Modules
12	Remote Master, Slice Master
18	High Speed Counter, Magnetic Pulse Input
20	8 pt Output

Code (Hex)	Component Type
21	8 pt Input
28	16 pt Output, F4 series Analog Output
2B	16 pt Input, F4 series Analog Input, Interrupt
30	32 pt Output, DL series Analog Output
3F	32 pt Input, DL series Analog Output
7F	Abnormal
FF	No module detected

Error Message Tables

The D4-454 CPUs will automatically log any system error codes and any custom messages you have created in your application program with the Fault instructions. (See Chapter 5 for details on the Fault instruction.) The CPU logs the error code, the date, and the time the error occurred. There are two separate tables that store this information.

System Error Table – stores up to 32 errors in the table

Fault Message Table – stores up to 16 messages in the table

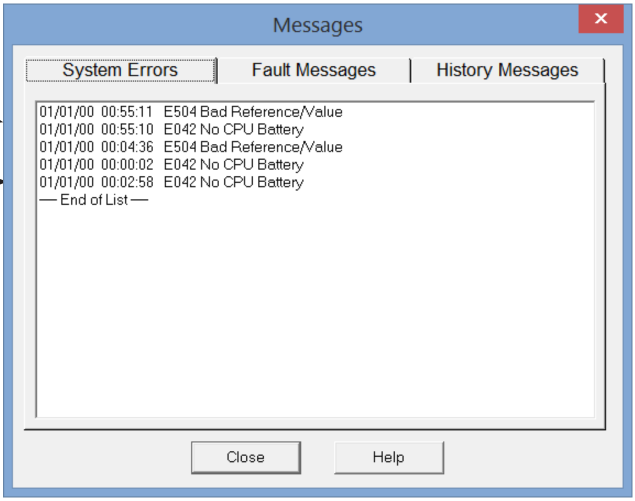
When an error or message is triggered, it is put into the first available table location. Therefore, the most recent error message may not appear in the first row of the table. If the table is full when an error occurs, the oldest error is pushed (erased) from the table and the new error is inserted in the row.

The following diagram shows an example of the Fault Message table as shown in DirectSOFT. You can access the error code table and the message table through DirectSOFT's PLC Diagnostic sub-menus. Details on how to access these logs are provided in the DirectSOFT manual.

Error Msg Example

Most recent message appears here, not at the top of the table.

Next message will show up in this row, which is now the oldest message.



```
01/01/00  00:55:11  E504 Bad Reference/Value
01/01/00  00:55:10  E042 No CPU Battery
01/01/00  00:04:36  E504 Bad Reference/Value
01/01/00  00:00:02  E042 No CPU Battery
01/01/00  00:02:58  E042 No CPU Battery
— End of List —
```

Program Error Codes

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 D4-454 system generates.

The following errors are captured in the System error log upon first detection or re-occurrence.

Error Code	Description
E003	Software time-out
E004	Invalid instruction(RAM parity error in the CPU)
E041	CPU battery low
E099	Program memory exceeded
E104	Write Fail
E151	Invalid Command

Error Code	Description
E201	Terminal block missing
E202	Missing I/O module
E203	Blown Fuse
E206	User 24V power supply failure
E250	Communication failure in the I/O chain
E251	I/O parity error
E252	New I/O configuration

These error are captured in the System error log if they exist when the CPU attempts to transition to RUN mode.

Error Code	Description
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

Error Code	Description
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

CPU Status Indicators

The D4-454 PLCs have indicators on the front to help you determine potential problems with the system. In normal runtime operation only, the RUN and PWR indicators are on. The table below is a quick reference to potential problems.

Indicator	Status	Potential Problems
PWR	OFF	1. Power input voltage is incorrect for selected operating mode, 110/220 VAC select jumper incorrect on CPU terminal strip 2. External power is off or disconnected (check fuses, breakers) 3. Power supply/CPU is faulty 4. Other component such as an I/O module has power supply shorted 5. Power budget exceeded for the CPU being used.
RUN	OFF	1. CPU programming error 2. Key switch in Stop position
	Flashing	CPU is in firmware upgrade mode
CPU	ON	1. Electrical Noise interference 2. CPU defective
BATT	Flashing	Battery voltage is low or Battery is not installed (V7745.12 must be On)
DIAG	ON	1. The CPU internal diagnostics has failed 2. The local bus on the backplane has had a communications error
I/O	ON	1. I/O module failure 2. External power supply failure 3. Configuration error 4. Base expansion unit failure
TXD	OFF	1. The CPU is not transmitting data on the second ports (ports 1, 2, and 3), due to programming error 2. The CPU is not in RUN mode
RXD	OFF	1. External device is not transmitting to CPU secondary ports (ports 1, 2, and 3) 2. Communications cable is defective, or not connected

PWR Indicator

In general there are four reasons for the CPU power status LED (PWR) to be OFF:

1. Power to the unit is incorrect or is not applied.
2. PLC power supply is faulty.
3. Other component(s) have the power supply shut down.
4. Power budget (+5V) for the CPU has been exceeded.

If the voltage to the power supply is not correct, the PLC 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.

5. First, disconnect the external power.
6. Verify that all external circuit breakers or fuses are still intact.
7. Check all incoming wiring for loose connections. If you're using a separate termination block, check those connections for accuracy and integrity.
8. If the connections are acceptable, reconnect the system power and verify the voltage at the D4-454 power input is within specification. If the voltage is not correct, shut down the system and correct the problem.
9. If all wiring is connected correctly and the incoming power is within the specifications, the PLC internal supply may be faulty.

If the voltage to the power supply is not correct, the CPU may not operate properly, if at all. For a new installation on AC-powered CPU versions, first check the 110/220 VAC select jumper on the terminal strip of the CPU. If the 110 VAC selection shunt is not installed while using 110 VAC, you will see the following symptoms:

- The communication ports will not function
- The CPU will only operate when no modules are installed

If the 110 VAC selection shunt is installed while using 220 VAC, the power supply in the CPU will be damaged. If this has happened, you will need to replace the CPU. The best way to check for a faulty CPU power supply is to substitute a known good one to see if this corrects the problem.

If the jumper is correctly installed for the AC version you are using, then measure the voltage at the terminal strip to ensure it is within the CPU input specs.

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 top port on the CPU. To test for a device causing this problem:

- Turn off power to the CPU.
- Disconnect all external devices (example communication cables) from the CPU.
- Reapply power to the system.

If the power supply operates normally, you probably 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:

Turn off power to the CPU.

Remove the CPU from the base, leaving its power cord attached.

Reapply power to the CPU.

If the PWR LED operates normally, the problem is most likely in one of the modules in the local CPU base. To isolate which module is causing the problem, remove one module at a time until the PWR LED operates normally. Put the CPU back in the base prior to testing for a bad module. Follow the procedure below:

- Turn off power to the CPU.
- Remove a module from the base.
- Reapply power to the CPU.

Bent base connector pins on the module can cause this problem, so check the connector. Remember that exceeding the power budget is a common error that will cause the PWR indicator to not come on or to come on intermittently.

Power budgeting problems usually appear during system start-up rather than after a long period of operation. If there is any doubt, it's a good idea to recheck this.

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, Systems Design and 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 software to determine the cause of the error:

- If you are attempting to enter Run mode with DirectSOFT, make sure the CPU mode switch is in the TERM position and try to enter Run mode.
- If you are using the CPU mode switch to change Run mode and the CPU does not respond, use DirectSOFT to diagnose the problem.
- If the RUN indicator is flashing (blinking), the CPU is in the firmware upgrade mode. You will need to update the firmware using KOYO firmware update tool, available for download on the AutomationDirect website.

Both of the programming devices, DirectSOFT programming software and a Handheld Programmer, will return an error message describing the problem. Depending on the error, there may also be an error code 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 the Error Codes Appendix.

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

The BATT indicator needs to be enabled with bit V7745.12 = 1. The BATT indicator will flash when the CPU battery voltage is low (2.5V or less) or the battery is not installed in the battery holder. The battery voltage is continuously monitored while the system voltage is being supplied. 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. When the BATT indicator is enabled, special relay SP43 comes on when the battery voltage is low and V7746 indicates the battery voltage in tenths of a volt.

DIAG Indicator

The diagnostics indicator is normally off. It turns on if the CPU detects a failure of its run-time diagnostics. Extreme electrical noise may cause a diagnostics failure, so power cycle the CPU first. If the DIAG indicator still turns on, the CPU is probably faulty. Replace it with a known good one to be sure.

I/O Indicator

If this indicator is on, a problem in the local, expansion, or remote I/O chain has been detected. Any of the problems listed below could be the cause of the I/O LED being on:

- A blown fuse inside an I/O module
- A loose terminal block
- The 24 VDC supply has failed
- The module or Expansion unit has failed
- The I/O configuration check detects a change in the I/O configuration

To aid you in further diagnosing where the I/O error is, each I/O module has LEDs to indicate if an error is present. The discrete I/O modules covered in this manual may have a combination of the following I/O indicators:

Indicator	Error Condition
TB	Loose or missing terminal block
24V	External 24V power supply not providing the correct voltage
FU	Module fuse has blown (check the I/O modules specification sheets to see if the fuse is replaceable)

Many other specialty modules also have indicators. The manuals for those products contain information on the indicators and status LEDs. If the modules are not providing any clues to the problem, run the I/O diagnostics in the DirectSOFT programming software. This 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.

TXD and RXD Indicators

The TXD and RXD indicators on the D4-454 CPU turn ON whenever the CPU either transmits or receives data, respectively. If the indicator(s) remain off when you are expecting communications, there is a problem. If the indicator(s) remain on when you are not expecting communications, there is a problem.

The TXD and RXD indicators turn on when data is transmitted on port 1, 2, and 3 of the D4-454. Therefore, when DirectSOFT programming software or an operator interface such as the DV-1000 is connected, the TXD and RXD are on constantly. If you are trying to detect communications originated by the ladder program itself, it may be useful to disconnect the programing device or operator interface. In this way, only the cable for the communications you are debugging is connected.

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. Make sure link settings match the PLC port settings.
- The device connected to the port is sending data incorrectly, or another application is running on the device.
- A grounding difference exists between the two devices.
- Electrical noise is causing intermittent errors.
- The PLC has a bad communication port and should be replaced.

For problems in communicating with DirectSOFT programming software on a personal computer, refer to the DirectSOFT programming user manual. It includes a troubleshooting section that can help you diagnose PC problems in communications port setup, address or interrupt conflicts, etc.

I/O Point Troubleshooting

Possible Causes

If you suspect an I/O error, there are several things that could be causing the problem.

- I/O configuration error on modules such as: Analog I/O, High Speed Counters, Specialized Communications, etc.
- A blown fuse in your machine or panel
- A loose terminal block
- The auxiliary 24 VDC supply has failed
- The Input or Output Point has failed

Some Quick Steps

When troubleshooting the DL405 PLCs, please be aware of the following facts which may assist you in quickly correcting an I/O problem.

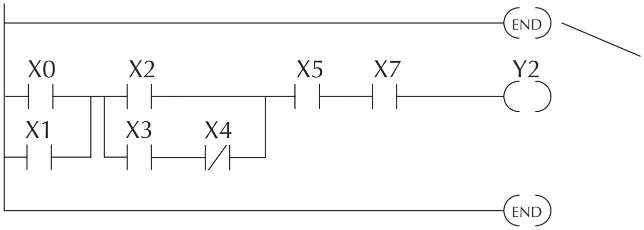
- The output modules cannot detect shorted or open output points. If you suspect one or more faulty points, measure the voltage drop from the common to the suspect point. Remember when using a Digital Volt Meter, leakage current from an output device such as a triac or a transistor must be considered. A point which is off may appear to be on if no load is connected the point.
- The I/O point status indicators 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 point the status indicators could be operating normally while the actual output device (transistor, triac etc.) could be damaged. With an input point, if the indicator LED is on, the input circuitry is probably operating properly. Verify the LED goes off when the input signal is removed.
- Leakage current can be a problem when connecting field devices to an I/O point. 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 10K to 20K resistor will work. Verify the wattage rating of the resistor is correct for your application.
- Because of the removable terminal blocks on the D4-454, the easiest method to determine if an I/O circuit has failed is to replace the unit if you have a spare. However, if you suspect a field device is defective, that device may cause the same failure in the replacement PLC as well. As a point of caution, you may want to check devices or power supplies connected to the failed I/O circuit before replacing the unit with a spare.
- The fuse blown indicator on an output module will indicate a problem only if an output point is connected to a load and the point is turned on. This indicator works by sensing a voltage drop across the fuse so there must be a voltage applied to the fuse and load applied to output to create the voltage drop before it can be reported by the module.

Testing Output Points

Output points can be set on or off in the D4-454 CPUs. If you want to do an I/O check-out independent of the application program, follow the procedure below:

Step	Action
1	Change the CPU mode switch to TERM.
2	Use DirectSOFT programming software to communicate online to the PLC.
3	Change to Program Mode.
4	Go to address 0.
5	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).
6	Change to Run Mode.
7	Using dataview to set (turn) on or off the points you wish to test. *
8	When you finish testing I/O points delete the “END” statement at address 0.
*** For more information refer to PC-DSOFT6-M, DirectSOFT programming software.**	

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.



Insert an END statement at the beginning of the program. This disables the remainder of the program.

Noise Troubleshooting

Electrical Noise Problems

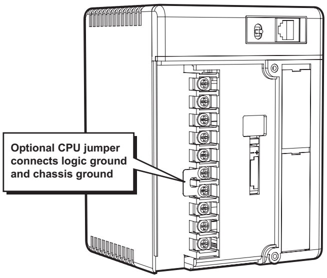
Noise is one of the most difficult problems to diagnose. Electrical noise can enter a system in many different ways and falls into one of two categories, conducted or radiated. It may be difficult to determine how the noise is entering the system but the corrective actions for either of the types of noise problems 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 circuit, 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 can act as a large antenna, introducing noise into the system, so, 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 PLC and I/O circuits. Installing an isolation transformer for all AC sources can correct this problem. DC sources should be properly grounded, except for Class II power supplies. Switching DC power supplies commonly generate more noise than linear supplies.
- Place input and output wiring in separate wireways or wiring bundles. Keep AC and DC wiring separated as well. Never run I/O wiring parallel or close proximity to high voltage wiring.
- To improve noise immunity, you may optionally install the factory provided shunt between logic ground (LG) and chassis ground (G) on the CPU terminal strip to the left.


Optional CPU jumper connects logic ground and chassis ground

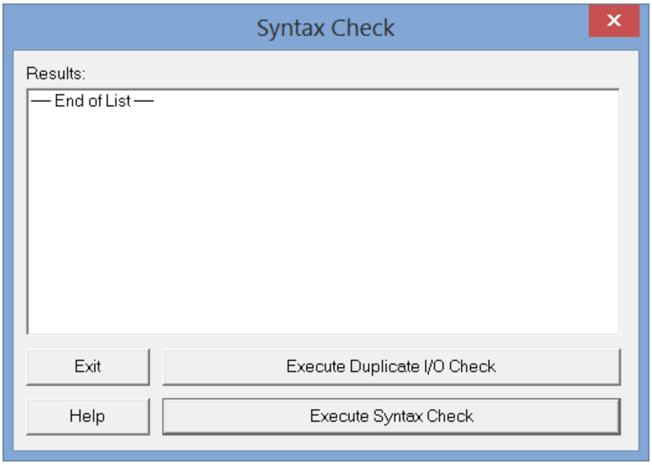
Machine Startup and Program Troubleshooting

The DL405 PLCs provide several features that can 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
- Special Instructions
- Run Time Edits
- Forcing I/O Points

Syntax Check

Even though DirectSOFT programming software provides error checking during program entry, you may want to check a program that has been modified. For example, you can use the PLC Diagnostics menu option within DirectSOFT to select Syntax Check (PLC -> Diagnostics -> Syntax Check). This check will find a wide variety of programming errors.



— End of List —

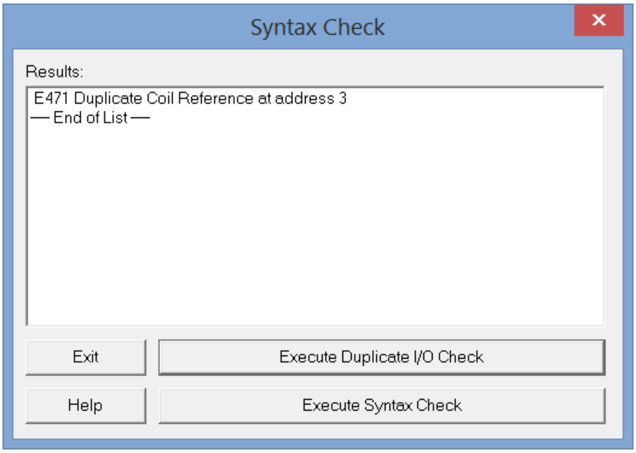
Diagnostic Window when no syntax error is found:

If you get an error, see the Error Codes Section for a complete listing of programming error codes. Correct the problem and continue running the Syntax check until the End of List message appears.

Duplicate I/O Check

You can check for multiple uses of the same output coil. For example, you can use the PLC Diagnostics menu option within DirectSOFT to select Execute Duplicate I/O Check (PLC -> Diagnostics -> Execute Duplicate I/O Check).

Diagnostic Window indicating a duplicate coil was found:



```
E471 Duplicate Coil Reference at address 3
— End of List —
```

If you get a Duplicate Reference error, correct the problem and continue running the Duplicate Reference check until no duplicate references are found.

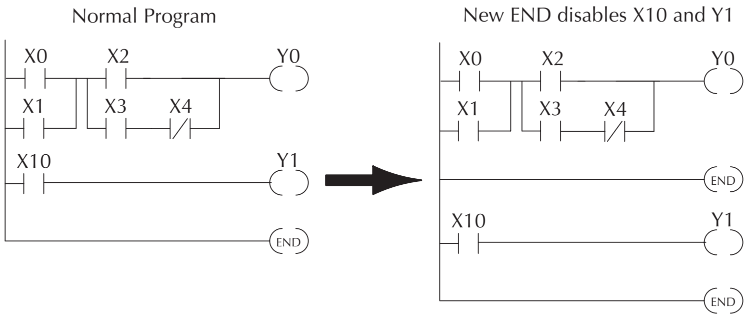
NOTE: *You can use the same coil in more than one location, especially in programs that use the Stage instructions and / or the OROUT instructions. The Duplicate Reference check will find these outputs even though they may be used in an acceptable fashion.*

Special Debug Instructions

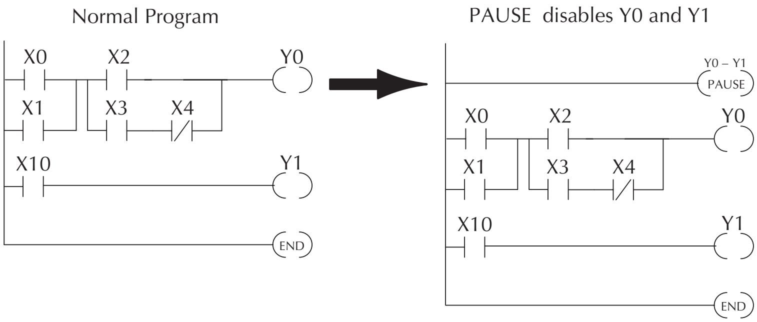
There are several instructions that can be used to help you debug your program during machine startup operations.

- END
- PAUSE
- STOP

END Instruction: If you need a way to quickly disable part of the program, just insert an END statement prior to the portion that should be disabled. When the CPU encounters the END statement, it assumes that is the end of the program. The following diagram shows an example.

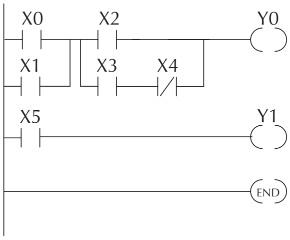


PAUSE Instruction: This instruction provides a quick way to allow the inputs (or other logic) to operate while disabling selected outputs. The output image register is still updated, but the output circuits are not. For example, you could make this conditional by adding an input contact or CR to control the instruction with a switch or a programming device. Or, you could just add the instruction without any conditions so the selected outputs would be disabled at all times.

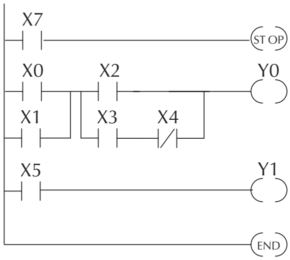


STOP Instruction: Sometimes during machine startup you need a way to quickly turn off all the outputs and return to Program Mode. You can use the STOP instruction. When this instruction is executed the CPU automatically exits Run Mode and enters Program Mode. Remember, all outputs are turned off during Program Mode. The following diagram shows an example of a condition that returns the CPU to Program Mode.

Normal Program



STOP puts CPU in Program Mode



In the example shown above, you could trigger X7 which would execute the STOP instruction. The CPU would enter Program Mode and all outputs would be turned off.

Run Time Edits

The D4-454 CPU allows you to make changes to the application program during Run Mode. These edits are not "bump-less." 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 operational 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 (Boolean)
AND, ANDN	And, And not (Boolean)
OR, ORN	Or, Or not (Boolean)
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 (Comparative Boolean)
AND, ANDN	And greater than or equal And less than (Comparative Boolean)

Mnemonic	Description
OR, ORN	Or greater than or equal or less than (Comparative Boolean)
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

Forcing I/O Points

There are many times, especially during machine startup and troubleshooting, that 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 D4-454 CPUs process the forcing requests.

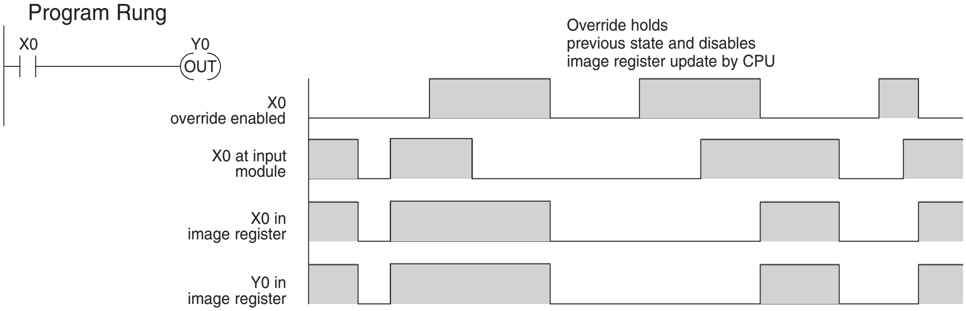
WARNING: Only authorized personnel fully familiar with all aspects of the application should make changes to the program. Make sure you thoroughly consider the impact of any changes to minimize the risk of personal injury or damage to equipment.

There are two types of forcing available with the D4-454 CPUs. (Chapter 3 provides a detailed description of how the CPU processes each type of forcing request).

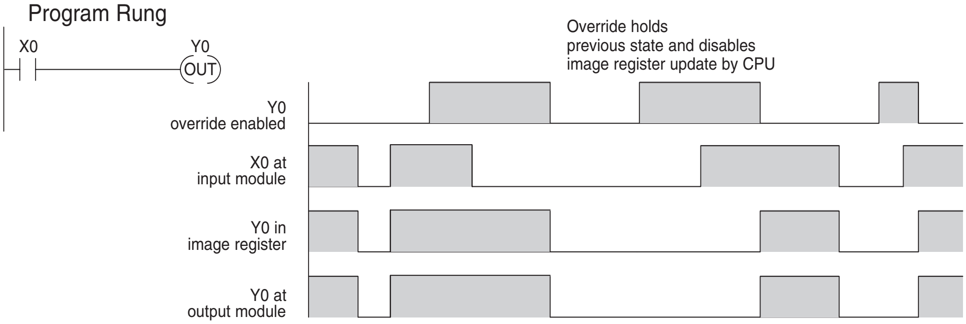
- **Regular Forcing:** This type of forcing can temporarily change the status of a discrete bit. For example, you may want to force an input on, even though it is really off. This allows you to change the point status that was stored in the image register. This value will be valid until the image register 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:** Bit override can be enabled by a menu option in the DirectSOFT programming software.. 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.

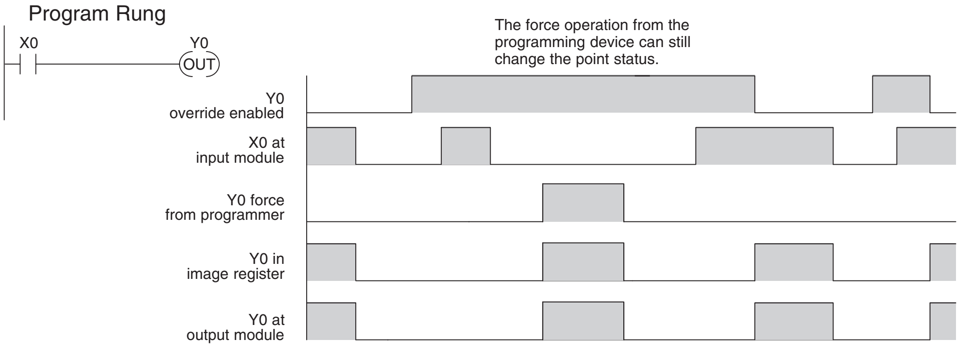
The following diagrams show how the bit override works for both input and output points. The example uses a simple rung, but the concepts are similar for any type of bit memory.



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.



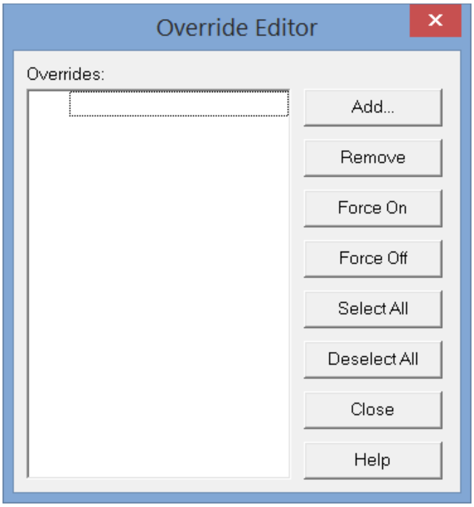
The following diagram shows how you can use a programming device in combination with the bit override to change the status of the point. Remember, bit override only disables CPU changes. You can still use a programming device to force the status of the point. Plus, since bit override maintains the current status, this enables true forcing. The example shown is for an output point, but you can also use the other bit data types.



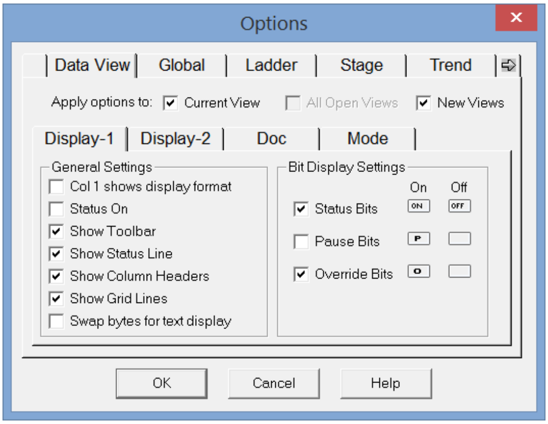
Bit Override Forcing

It is possible to Force a bit On/Off by using the Override Editor or using Data View. Remember that when 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 override editor is found in the PLC menu. Select PLC, Setup and Overrides. You can enter the desired bit(s) to override in the editor by pressing the Add button. To force a bit, select the bit in the list, select the Force On button and confirm the action. Follow the same steps to Force Off and to Remove.



To override a bit using Data View, first open a data window from the Debug menu. Select Debug, Data View, New. Type the bit address under the Element column of the Data window. If the Edits column is not available, click on the Select Mode button at the top left of the Data View window to view the Override Bits buttons. It may be necessary to expand the Data View windows in order for the buttons to be visible.



Apply options to: ☑ Current View □ All Open Views ☑ New Views

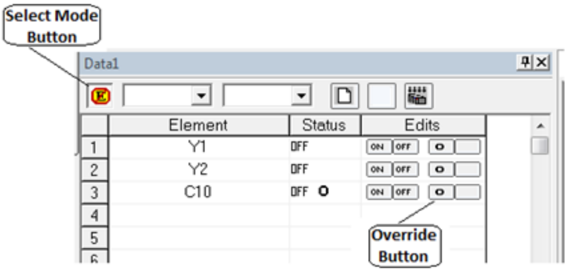
General Settings
□ Col 1 shows display format
□ Status On
☑ Show Toolbar
☑ Show Status Line
☑ Show Column Headers
☑ Show Grid Lines
□ Swap bytes for text display

Bit Display Settings

	On	Off
☑ Status Bits	ON	OFF
□ Pause Bits	P	
☑ Override Bits	O	

If the Override buttons do not show up in the Data View window after adding a bit in the Element column, use the Options dialog to enable the Override bits. Right click on the Data View window and select Options from the list. The Options dialog will be displayed with the Data View tab in view. Select the Override Bits under the Bit Display Settings and the Override buttons will become available in the Data View window.

To set an Override Bit, double-click on the respective Override button with the letter “O”. Double-clicking on the Blank button will turn the Override feature Off. The letter “O” in the status column indicates the Override Bit is set for that bit.



	Element	Status	Edits
1	Y1	OFF	ON OFF O
2	Y2	OFF	ON OFF O
3	C10	OFF O	ON OFF O
4			
5			
6			

***NOTE:** For more information on using and setting up Data View, refer to the DirectSOFT programming software user manual, PC-DSOFT6-M, Chapter 10. A PDF copy of the DirectSOFT user manual is available in the Help folder under the DirectSOFT 6 installation folder.*

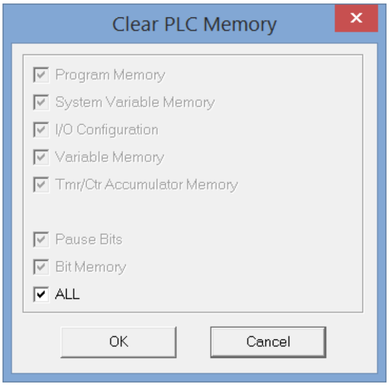
Reset the PLC to Factory Defaults

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".

Step Two – While connected with DirectSOFT, go the PLC menu and then to the "Setup" submenu and select; "Initialize Scratch Pad" and press "Ok".

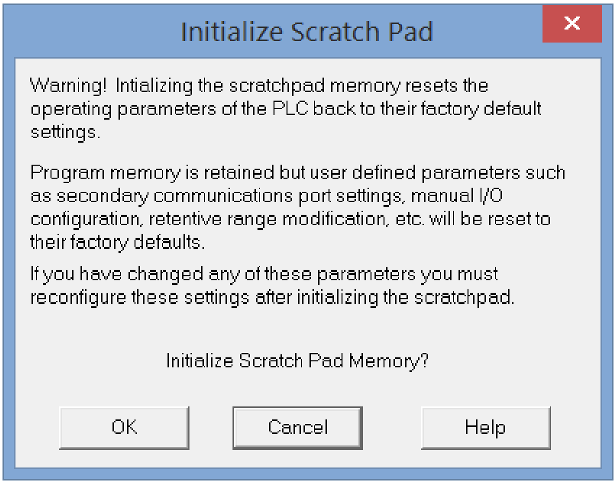


Clear PLC Memory

☑ Program Memory
☑ System Variable Memory
☑ I/O Configuration
☑ Variable Memory
☑ Tmr/Ctr Accumulator Memory

☑ Pause Bits
☑ Bit Memory
☑ ALL

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 IO will be reset to factory default settings.



Initialize Scratch Pad

Warning! Intializing the scratchpad memory resets the operating parameters of the PLC back to their factory default settings.

Program memory is retained but user defined parameters such as secondary communications port settings, manual I/O configuration, retentive range modification, etc. will be reset to their factory defaults.

If you have changed any of these parameters you must reconfigure these settings after initializing the scratchpad.

Initialize Scratch Pad Memory?

The PLC has now been reset to factory defaults and you can proceed to program the PLC.

D4-454 Error Codes

Appendix A

In This Appendix...

D4-454 Error Codes

D4-454 Error Code	Description
E001 CPU FATAL ERROR	You may possibly clear the error by power cycling the CPU. If the error returns, replace the CPU.
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 extend the time alloted to the watchdog timer using PLC Setup menu in ***Direct***SOFT programming software.
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. The CPU indicator will blink if the battery is less than 2.5 VDC (refer to the table on page 3-5).
E042 NO CPU BATTERY	The CPU battery is not installed. 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. Power cycle the CPU. If the error returns, 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 CPU.
E2** I/O MODULE FAILURE	An I/O module has failed. Using ***Direct***SOFT programming software, from the PLC menu, run Diagnostics -> I/O Diagnostics to determine the actual error.
E201 TERMINAL BLOCK MISSING	A terminal block is loose or missing from an I/O module. SP45 will be on and the error code will be stored in V7756.
E202 MISSING I/O MODULE	An I/O module has failed to communicate with the CPU or is missing from the slot. SP45 will be on and the error code will be stored in V7756. Using ***Direct***SOFT programming software, from the PLC menu, run Diagnostics -> I/O Diagnostics to determine the slot and base location of the module reporting the error.
E203 BLOWN FUSE	A fuse has blown in an I/O module. SP45 will be on and the error code will be stored in V7756. Using ***Direct***SOFT programming software, from the PLC menu, run Diagnostics -> I/O Diagnostics to determine the slot and base location of the module reporting the error.
E206 USER 24V POWER SUPPLY FAILURE	The 24 VDC power supply being used to power output modules has failed. 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 CPU.
E250 COMM FAILURE IN THE I/O CHAIN	A failure has occurred in the I/O system. The problem could be in the base, expansion cable or I/O Expansion Unit power supply. Check all cabling between bases and replace faulty hardware, if necessary. SP45 will be on and the error code will be stored in V7755. Using ***Direct***SOFT programming software, from the PLC menu, run Diagnostics -> I/O Diagnostics to determine the base location reporting the error.
E251 I/O PARITY ERROR	A communication parity error has occurred in the I/O communication chain.
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 Using ***Direct***SOFT programming software, from the PLC menu, run Diagnostics -> I/O Diagnostics to accept the new configuration. SP45 will be on and the error code will be stored in V7755.
E261 I/O ADDRESS CONFLICT	Overlapping addresses have been assigned while manually configuring the I/O. Correct the address assignments using ***Direct***SOFT programming software, from the PLC menu, select the Configure I/O menu option. SP45 will be on and the error code will be stored in V7755.

D4-454 Error Code	Description
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.
E263 CONFIGURED I/O ADDRESS OUT OF RANGE	Out of range addresses have been assigned while manually configuring the I/O. Correct the address assignments using ***Direct***SOFT programming software PLC menu, run Diagnostics -> I/O Diagnostics.
E264 DUPLICATE I/O REFERENCE	Duplicate addresses have been assigned while manually configuring the I/O. Correct the address assignments using ***Direct***SOFT programming software, from the PLC menu, select Configure I/O menu option.
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.
E352 BACKGROUND 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. If the error continues, replace the CPU.
E4** NO PROGRAM	A syntax error exists in the application program. The most common is a missing END statement. Using ***Direct***SOFT programming software, from the PLC Diagnostics menu option, select System Information or Syntax Check 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 appropriate location in your program. SP52 will be on and the error code will be stored in V7755. A program with the end statement included should be downloaded to PLC and PLC placed into RUN mode before error will clear.
E402 MISSING LBL	A GOTO, GTS, MOVMC, or LDLBL instruction was used without the appropriate label. Refer to Chapter 5 for details on these instructions. SP52 will be on and the error code will be stored in V7755.
E403 MISSING RET	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	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	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>256	There is greater than 256 SBR or DLBL instructions in the program. This error is also returned if there is greater than 256 GOTO instructions used in the 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 LBL REFERENCE	Two or more LBL instructions exist in the application program with the same number. A unique number must be allowed for each label. SP52 will be on and the error code will be stored in V7755.
E423 NESTED LOOPS	Nested loops (programming one FOR/NEXT loop inside of another) are not allowed. SP52 will be on and the error code will be stored in V7755.
E431 INVALID ISG/SG ADDRESS	An ISG or SG instruction must not be placed after the end statement (such as inside a subroutine). SP52 will be on and the error code will be stored in V7755.
E432 INVALID JUMP (GOTO) ADDRESS	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.

D4-454 Error Code	Description
E433 INVALID SBR ADDRESS	An 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.
E434 INVALID RTC ADDRESS	An RTC 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	An 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.
E437 INVALID IRTC ADDRESS	An IRTC 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	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.
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/UDC	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	An STR/STRN 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.

D4-454 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 INSTRUCTION	An instruction exists between convergence stages.
E482 MAX. CV INSTRUCTIONS EXCEEDED	Number of CV instructions exceeds 17.
E483 INVALID CV JUMP 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 MISSING REQUIRED INSTRUCTION	A CV JMP instruction is not placed between the CV and the [SG, ISG, ST BLK, END BLK, END] instruction.
E486 INVALID CALL BLK ADDRESS	CALL BLK is used in a subroutine or a program interrupt routine. The CALL BLK instruction may only be used in the main program area (before the END statement).
E487 MISSING ST BLK INSTRUCTION	The CALL BLK instruction is not followed by a ST BLK instruction.
E488 INVALID ST BLK ADDRESS	The ST BLK instruction is used in a subroutine or a program interrupt. Another ST BLK instruction is used between the CALL BLK and the END BLK instructions.
E489 DUPLICATE 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.
E491 INVALID ISG INSTRUCTION ADDRESS	There is an ISG instruction between the ST BLK and END BLK instructions.
E492 INVALID END BLK ADDRESS	The END BLK instruction is used in a subroutine or a program interrupt routine. The END BLK instruction is not followed by a ST BLK instruction.
E493 MISSING END REQUIRED INSTRUCTION	A [CV, SG, ISG, ST BLK, END] instruction must immediately follow the END BLK instruction.
E494 MISSING END BLK INSTRUCTION	The ST BLK instruction is not followed by a END BLK instruction.
E499 PRINT INSTRUCTION	Invalid PRINT instruction 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 command was entered into the handheld programmer.
E504 BAD REF/VAL	An invalid value or reference number was entered with an instruction.

D4-454 Error Code	Description
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.
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	An operation was attempted by the handheld programmer or ***Direct***SOFT programming software 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.
E540 CPU LOCKED	The CPU has been password locked. To unlock the CPU, enter the password when prompted by ***Direct***SOFT programming software. .
E541 WRONG PASSWORD	The password used to unlock the CPU ***Direct***SOFT programming software 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 ***Direct***SOFT programming software PLC Password menu option.
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.
E603 DATA MISSING	A search function was performed and the data was not found.
E604 REFERENCE MISSING	A search function was performed and the reference was not found.
E610 BAD I/O TYPE	The application program has referenced an I/O module as the incorrect type of module.

Appendix B

Special Relays

In This Appendix...

D4-454 PLC Special Relays

Special Relays are just contacts that are set by the CPU operating system to indicate a particular system event has occurred. These contacts are available for use in your ladder program. Knowing just the right special relay contact to use for a particular situation can save a lot of programming time. Since the CPU operating system sets and clears special relay contacts, the ladder program only has to use them as inputs in ladder logic.

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 ensure 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 50 ms. and off for 50 ms.
SP6	50 ms clock	On for 25 ms. and off for 25 ms.
SP7	Alternate scan	On every other scan.

CPU Status Relays

SP11	Forced run mode	On when the mode switch is in the run position and the CPU is running.
SP12	Terminal run mode	On when the mode switch is in the TERM position and the CPU is in the run mode.
SP13	Test run mode	On when the CPU is in the test run mode.
SP15	Test stop mode	On when the CPU is in the test stop mode.
SP16	Terminal PGM mode	On when the mode switch is in the TERM position and the CPU is in program mode.
SP17	Forced stop	On when the mode switch is in the STOP position.
SP21	Break relay 2	On when the BREAK instruction is executed. It is off only 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	On when hardware interrupts are enabled using the ENI instruction.
SP26	I/O update disable	Turned on by the application program, thus freezing the state of the I/O.
SP27	Selectable I/O update disable	Relay can be turned on by the application program. If turned on the I/O update is frozen at last state, for I/O modules sensing a missing terminal block.
SP37	Scan control error	This relay will be on if the actual scan time is in excess of the scan time set in fixed or limit scan modes. The relay contact may be useful in program error recovery.

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	Low battery error	On when the CPU battery voltage is low (only if bit 12 of V7745 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 such as a blown fuse or missing 24V has occurred.
SP46	Communications error	On when a communication error occurs on any of the CPU ports.
SP47	I/O configuration error	On if an I/O configuration error has occurred. The I/O configuration check must be set on in the CPU 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	Math/Table pointer error	On if there is a math execution error or a table pointer error.
SP54	Communication error	On when RX, WX,RD, WT, instructions are executed with the wrong parameters.
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

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	Pointer reference error	On when the V-memory specified by a pointer (P) is not valid.
SP72	Floating point number	On anytime the value in the accumulator is a valid 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	Underflow	On anytime a floating point 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.

Communication Monitoring Relay

The Communications Monitoring Relays are numbered in pairs corresponding to CPU port numbers or slot positions in a base. The two relay types are:

Module/port busy – on when the port or the communication module in the referenced slot and base is busy transmitting or receiving. You must use this relay with RX and WX instructions to prevent attempting to execute a RX or WX while the modules is busy.

Communication error – on when a communications error occurs. This error automatically clears when another RX or WX instruction executes.

All D4-454 CPUs							
		SP112	CPU port busy Port 1	SP114	CPU port busy Port 2	SP116	CPU port busy Port 3
		SP113	Comm error Port 1	SP115	Comm error Port 2	SP117	Comm error Port 3
Local Base		Expansion Base #1		Expansion Base #2		Expansion Base #3	
SP120	Module busy Slot 0	SP140	Module Busy Slot 0	SP160	Module busy Slot 0	SP200	Module busy Slot 0
SP121	Comm error Slot 0	SP141	Comm error Slot 0	SP161	Comm error Slot 0	SP201	Comm error Slot 0
SP122	Module busy Slot 1	SP142	Module busy Slot 1	SP162	Module busy Slot 1	SP202	Module busy Slot 1
SP123	Comm error Slot 1	SP143	Comm error Slot 1	SP163	Comm error Slot 1	SP203	Comm error Slot 1
SP124	Module busy Slot 2	SP144	Module busy Slot 2	SP164	Module busy Slot 2	SP204	Module busy Slot 2
SP125	Comm error Slot 2	SP145	Comm error Slot 2	SP165	Comm error Slot 2	SP205	Comm error Slot 2
SP126	Module busy Slot 3	SP146	Module busy Slot 3	SP166	Module busy Slot 3	SP206	Module busy Slot 3
SP127	Comm error Slot 3	SP147	Comm error Slot 3	SP167	Comm error Slot 3	SP207	Comm error Slot 3
SP130	Module busy Slot 4	SP150	Module busy Slot 4	SP170	Module busy Slot 4	SP210	Module busy Slot 4
SP131	Comm error Slot 4	SP151	Comm error Slot 4	SP171	Comm error Slot 4	SP211	Comm error Slot 4
SP132	Module busy Slot 5	SP152	Module busy Slot 5	SP172	Module busy Slot 5	SP212	Module busy Slot 5
SP133	Comm error Slot 5	SP153	Comm error Slot 5	SP173	Comm error Slot 5	SP213	Comm error Slot 5
SP134	Module busy Slot 6	SP154	Module busy Slot 6	SP174	Module busy Slot 6	SP214	Module busy Slot 6
SP135	Comm error Slot 6	SP155	Comm error Slot 6	SP175	Comm error Slot 6	SP215	Comm error Slot 6
SP136	Module busy Slot 7	SP156	Module busy Slot 7	SP176	Module busy Slot 7	SP216	Module busy Slot 7
SP137	Comm error Slot 7	SP157	Comm error Slot 7	SP177	Comm error Slot 7	SP217	Comm error Slot 7

Appendix C

Product Weights

In This Appendix...

Product Weight

Product Weight Table	
CPU	***Weight***
D4-454	25.2 oz (713g)
D4-454DC-1	26.0 oz (737g)
Base	***Weight***
D4-04B-1	23.3 oz (660g)
D4-06B-1	29.3 oz (830g)
D4-08B-1	34.9 oz (990g)
Expansion	***Weight***
D4-EX	22.7 oz (644g)
D4-EXDC	23.3 oz (660g)
DC Inputs	***Weight***
D4-08ND3S	8.8 oz (250g)
D4-16ND2	8.8 oz (250g)
D4-16ND2F	8.8 oz (250g)
D4-32ND3-1	6.7 oz (190g)
D4-64ND2	7.8 oz (220g)
AC Inputs	***Weight***
D4-08NA	8.5 oz (240g)
D4-16NA	9.5 oz (270g)
AC/DC Inputs	***Weight***
D4-16NE3	8.8 oz (250g)
F4-08NES	9.0 oz (256g)
DC Outputs	***Weight***
F4-08TD1S	9.9 oz (282g)
D4-16TD1	9.5 oz (270g)
D4-16TD2	9.5 oz (270g)
D4-32TD1	6.7 oz (190g)
D4-32TD1-1	6.7 oz (190g)
D4-32TD2	6.7 oz (190g)
D4-64TD1	7.4 oz (210g)

Product Weight Table	
AC Outputs	***Weight***
D4-08TA	11.6 oz (330g)
D4-16TA	12.3 oz (350g)
Analog	***Weight***
F4-04AD	10.6 oz (300g)
F4-04ADS	11.5 oz (326g)
F4-08AD	11.0 oz (312g)
F4-04DA-1	9.2 oz (262g)
F4-04DA-2	9.3 oz (264g)
F4-04DAS-1	9.9 oz (281g)
F4-04DAS-2	11.2 oz (317g)
F4-08DA-1	11.2 oz (318g)
F4-08DA-2	9.7 oz (274g)
F4-16AD-1	11.7 oz (331g)
F4-16AD-2	11.5 oz (325g)
F4-16DA-1	11.5 oz (326g)
F4-16DA-2	9.9 oz (280g)
F4-08THM	8.0 oz (227g)
F4-08THM-J	9.7 oz (276g)
F4-08THM-K	9.9 oz (280g)
F4-08THM-T	8.0 oz (227g)
F4-08RTD	9.7 oz (275g)
CoProcessors	***Weight***
F4-CP128-1	8.9 oz (252g)
F4-CP128-T	9.9 oz (281g)
Communications	***Weight***
D4-DCM	8.2 oz (233g)
Networking	***Weight***
F4-MAS-MB	8.9 oz (252g)
H4-ECOM100	6.4 oz (182g)

Product Weight Table	
Remote I/O	***Weight***
D4-RM	8.0 oz (228g)
D4-RS	27.1 oz (767g)
D4-RSDC	26.8 oz (760g)
H4-EBC	25.3oz (718g)
H4-ERM100	6.4oz (192g)
Specialty	***Weight***
F4-16PID)	7.3 oz (207g
F4-8MPI	12.3 oz (350g)
F4-4LTC	12.7 oz (361g)
H4-CTRIO	8.8 oz (250g)
D4-FILL	4.0 oz (357g

Appendix
D

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

Appendix E

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 for "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 familiar with 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. As with a computer, there are only two valid digits a PLC relies on, 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, as we are all well aware. 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 3x102 + 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 13x163 + 8x162 + 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 3x162 + 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 8 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

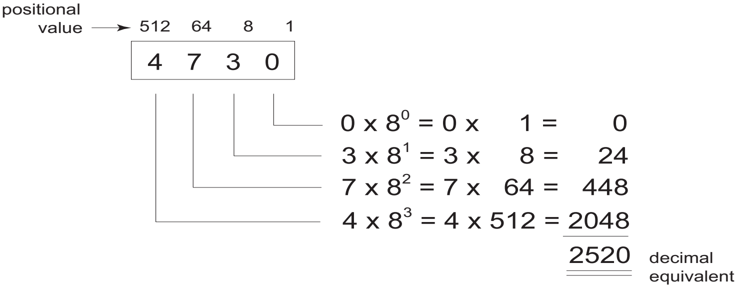
This follows the ***Direct***LOGIC PLCs. Refer to Chapter 3 bit maps and notice that the memory addresses are numbered in octal, as well as each bit. The octal system is much like counting in the decimal system without the digits 8 and 9 being available.

The general format for four digits of the octal number is:

$$(\mathrm{d}\ \mathrm{x}\ 8^{0}) + (\mathrm{d}\ \mathrm{x}\ 8^{1}) + (\mathrm{d}\ \mathrm{x}\ 8^{2}) + (\mathrm{d}\ \mathrm{x}\ 8^{3})$$

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, for example, 867 in BCD would be expressed the same as 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.

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

Most PLCs use the 32-bit format for floating point (or Real) numbers.

Floating point numbers which DirectLOGIC 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.

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 (0's and 1's), 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																
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) 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 (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 the 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 409510. 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 1653310. 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.

Base 10 Value	BCD Bit Pattern	Binary Bit Pattern
4095	0100 0000 1001 0101	1111 1111 1111

Table 9

Look at the following example and note the same value represented by the different numbering systems.

6 7	Decimal	4 6 6 0	Decimal
0110 0111	BCD	0100 0110 0110 0000	BCD
0100 0011	Binary	0001 0010 0011 0100	Binary
4 3	Hex	1 2 3 4	Hex
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 a positive number. Thus, a 16 bit word allows numbers in the range ±32767. Table 12 shows a representations 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. Without getting involved with a full explanation, a simple formula for two’s complement is to invert the binary and add one (see Table 12). Basically, 1’s are being changed to 0’s and all 0’s are being changed to 1.

Two’s Compliment	
Decimal	Binary
100	0000 0000 0110 0100
-100	1111 1111 1001 1100

Table 12

More information about 2’s complement can be found on the Internet.

AutomationDirect.com Products and Data Types

DirectLOGIC PLCs

The DirectLOGIC 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 programming, number conversions Intelligent Box (IBox) Instructions are available with DirectSOFT. These instruction descriptions can be found in the supplement manual DL405-IBOX-S in the math IBox group. Most DirectLOGIC analog modules can be setup 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. DirectLOGIC 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 DirectSOFT, be certain that the proper format is selected for the element to be viewed. The data type and length is 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".

Unsupported Hardware

Appendix F

In This Appendix...

Unsupported Hardware

Unsupported Modules Table*	
Bases	***Retired***
D4-04B, D4-04BNX	Yes
D4-06B, D4-06BNX	Yes
D4-08B, D4-08BNX	**Yes**
Input Modules	
D4-32ND3-2	Yes
D4-16NA-1	Yes
Output Modules	
D4-08TD	Yes
Analog Modules	
D4-04AD	Yes
Communications Modules	
H4-ECOM	**Yes**
H4-ECOM-F	–
Remote I/O Modules	
D4-ERM	Yes
D4-ERM-F	–
Specialty Modules	
D4-PULS	**Yes**
D4-INT	Yes
D4-HSC	Yes
F4-CP128-R	Yes
F4-CP512-1	**Yes**

***NOTE:** Any hardware with a date code less than 09X0 or with a first digit that is not 0, 1, or 2 may not work with the D4-454. We suggest that any hardware older than ten years and not currently sold on the AutomationDirect.com website, be upgraded to a newer version.

D4-450 and D4-454 Differences

D4-450 and D4-454 Differences		
Differences	***D4-450***	***D4-454***
Total Memory	22.8K/30.8K	46.8K
Ladder Memory	7.5K/15.5K*	31.5K
DirectSOFT	Yes	Yes, version 6.1 or later
Memory Cartridge	Yes	"Not needed (D4-454 has same amount of memory as the largest memory cartridge)
Battery	D3-D4-BAT	D2-BAT-1 (CR2354)
Mode Switch	Keyswitch	Toggle Switch (Same position/function)
Port 1 and 3 Baud rate	300, 600, 1200, 2400, 4800, 9600, 19200, 38400	2400, 4800, 9600, 19200, 38400
Port 1 and 3 settings	"8 data bits, 1 start bit, 1 stop bit, Odd, Even or No parity 7 data bits, 1 start bit, 1 stop bit, No parity"	8 data bits, 1 start bit, 1 stop bit, Odd, Even or No parity
Port 2 Protocol	DirectNet (master/slave), K-sequence, Non-procedure	"DirectNet (master/slave), K-sequence, Non-procedure, Modbus RTU (master/slave)"
Firmware Update	Port 1 only	Supported from all ports
Unsupported Hardware	–	See previous page

NOTE: *D4-454 will only support D4-HSC module firmware v2.3. D4-HSC modules with older firmware cannot be updated and are not compatible.*