Errata Sheet

This Errata Sheet contains corrections or changes made after the publication of this manual.

Product Family:	DL405	**Date**:	December 11, 2018
Manual Number	D4-USER-M		
Revision and Date	4th Edition, Rev. A; March 2013		

Changes to Chapter 2. Installation, Wiring, and Specifications

Page 2-4. Safety Guidelines; Class 1, Division 2 Approval

Delete this section, including the two warnings. The DL405 system is NOT Class 1, Division 2 approved.

Page 2-44. D4-32TD1-1, 5-15VDC Output

Add the following NOTE to the specifications table for this module:

NOTE:The D4-32TD1-1 module operates on reverse logic. Voltage is present when Output is Off; no voltage is present when output is On.

Changes to Chapter 4. System Design and Configuration

Page 4-8. Calculating the Power Budget; Module Power Requirements

Change the "5V Current Required (mA)" values in the table for these analog modules:

F4-04AD: Change from 85 mA to 150 mA

F4-04ADS: Change from 270 mA to 370 mA

Page 4-18. Network Connections to MODBUS and *Direct*NET; Configuring the CPU's Comm Ports

Change the NOTE near the bottom of the page to say:

NOTE: The recommended cable for RS-232 and RS-422 is AutomationDirect L19772-1 (Belden 8102) or equivalent.

Changes to Chapter 5. Standard RLL Instructions; Accumulator Logic Instructions

Page 5-85. Compare with Stack (CMPS)

The table below has more accurate descriptions for the discrete bit flags than the descriptions in the manual:

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.

Errata Sheet

This Errata Sheet contains corrections or changes made after the publication of this manual.

Changes to Chapter 8. PID Loop Operation (DL450 only)

Page 8-36. Monitor Limit Alarms. Add the following note at the end of this page:

NOTE: PID deviation alarm only works in Auto mode.

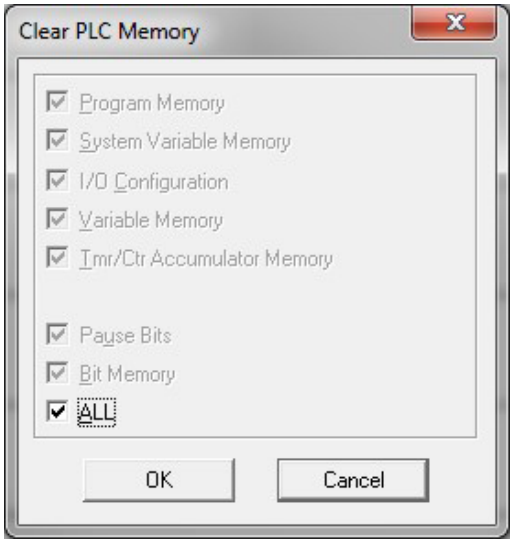
Changes to Chapter 9. Maintenance and Troubleshooting

Page 9-26. Add the following to the end of this chapter (right after BREAK Instruction placement):

Reset the PLC to Factory Defaults

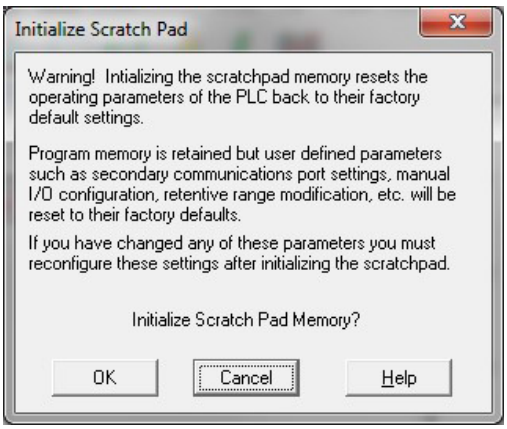
NOTE: Resetting to factory defaults will not clear any password stored in the PLC**.**

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 1 – 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 2 – While connected with DirectSoft, go the PLC menu and then to the "Setup" submenu and select "Initialize Scratch Pad". Press "Ok".



NOTE: 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: Retentive ranges will be reset to the factory settings.

NOTE: Manually addressed IO will be reset to factory default settings.

The PLC has now been reset to factory defaults and you can proceed to program the PLC.

Errata Sheet

This Errata Sheet contains corrections or changes made after the publication of this manual.

Changes to Appendix I. European Union Directives (CE)

Page I-4. Special Installation Manual

Delete this section. Manual D-EU-M is no longer available. Instead, refer to the websites listed below:

Page I-4: Other Sources of Information

Replace the text that follows the sentence "It may be possible to obtain this information locally; however, the official source of applicable Directives and related standards is:" with the following updated information:

Publications Office

2, rue Mercier

2985 Luxembourg

LUXEMBOURG

Quickest contact is via the web at:

http://ec.europa.eu/growth/single-market/european-standards/harmonised-standards

Another source is the British Standards Institution at:

British Standards Institution – Sales Department, Linford Wood:

Milton Keynes, MK14 6LE, United Kingdom.

The quickest contact is via the web at http://www.bsigroup.com

Another source is

The 'Blue Guide' on the implementation of EU product rules 2016

https://ec.europa.eu/commission/index_en

Page I-10. Items Specific to the DL405

Ignore the reference to manual D-EU-M in the first paragraph of this section. That manual is no longer available. See the websites listed above.

AUTOMATIONDIRECT.com

DL405 User Manual

Manual Number: D4-USER-M

WARNING

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

To minimize the risk of potential safety problems, you should follow all applicable local and national codes that regulate the installation and operation of your equipment. These codes vary from area to area and usually change with time. It is your responsibility to determine which codes should be followed, and to verify that the equipment, installation, and operation are 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 Desk Reference. If you have any questions concerning the installation or operation of this equipment, or if you need additional information, please call us at 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.

Copyright 2013, *Automationdirect.com* Incorporated
All Rights Reserved

No part of this manual shall be copied, reproduced, or transmitted in any way without the prior, written consent of ***Automationdirect.com*** **Incorporated**. **AutomationDirect** retains the exclusive rights to all information included in this document.

AVERTISSEMENT

Nous vous remercions d'avoir acheté l'équipement d'automatisation de **Automationdirect.com^mc** 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 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.

Copyright 2013, Automationdirect.com^mc Incorporated
Tous droits réservés
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.

Manual Revisions

Refer to this history in all correspondence and/or discussion about this manual.

Title: DL405 User Manual, 4th Edition, Rev. A

Manual Number: D4-USER-M

Issue	Date	Description of Changes
Original	1/94	Original Issue
2nd Edition	4/95	Major revision
3rd Edition	1/97	Major revision
Rev A	5/98	Minor corrections
Rev B	11/99	Minor corrections
Rev C	12/00	Added CE information, new PID features, minor corrections
Rev D	9/03	Added memory information, MODEM corrections, major and minor corrections
Rev E	7/04	Added ASCII table, minor corrections
4th Edition	8/09	Made corrections throughout the manual; updated Chapter 4 dialogs, revised Chapter 8; added Numbering Systems appendix.
	2/11	Manual reprinted with these 4 corrections: 1. Page 5-45: Replaced graphic for “Accumulator Timer Example Using Comparative Contacts”. 2. Page 5-80: Replaced graphic for “Exclusive OR Formatted (XORF) Example”. 3. Page 8-23: Made 2 changes to Bit 7 row in table: A. After “Autotune start”, added note saying “Bit 7 can be used to cancel Autotune mode by setting it to 0”. B. In fourth column, change “autotune done” to “autotune/cancel done”. 4. Appendix page I-5: Added enclosure graphic.
4th Edition	5/11	Corrected the number of memory regs needed in the print message instruction.
Rev A	3/13	Updated Inductive Load Suppression info.

Table of Contents

Chapter 1: Getting Started

Introduction **1-2**
The Purpose of this Manual 1-2
Where to Begin 1-2
Supplemental Manuals 1-2
Technical Support 1-2
Conventions Used **1-3**
Key Topics for Each Chapter 1-3
DL405 System Components **1-4**
CPUs 1-4
Bases 1-4
I/O Configuration 1-4
I/O Modules 1-4
Programming Methods **1-4**
DirectSOFT 5 Programming for Windows 1-4
Handheld Programmer 1-4
DL405 System Diagrams 1-5
***Direct*LOGIC Part Numbering System** **1-8**
Quick Start for PLC Checkout and Programming **1-10**
Steps to Designing a Successful System **1-14**
Frequently Asked Questions **1-16**

Chapter 2: Installation, Wiring, and Specifications

Safety Guidelines **2-2**
Plan for Safety 2-2
Three Levels of Protection 2-3
Emergency Stops 2-3
Emergency Power Disconnect 2-4
Orderly System Shutdown 2-4
Class 1, Division 2 Approval 2-4
Mounting Guidelines **2-5**
Base Dimensions 2-5
Panel Layout & Clearances 2-6
Enclosures 2-7
Agency Approvals 2-7
Environmental Specifications 2-8
Power 2-8
Component Dimensions 2-9
Installing DL405 Bases **2-10**
Three Sizes of Bases 2-10
Mounting the Base 2-10

Choosing the Base Type 2–11
Installing Components in the Base 2–12
Setting the CPU DIP Switches (DL430/440 Only) 2–12
CPU and Expansion Unit Wiring Guidelines 2–13
CPU Wiring 2–13
Expansion Unit Wiring 2–14
Connecting Programming Devices 2–14
Connecting Operator Interface Devices 2–15
I/O Wiring Strategies 2–16
PLC Isolation Boundaries 2–16
Powering I/O Circuits with the Auxiliary Supply 2–17
Powering I/O Circuits Using Separate Supplies 2–18
Sinking/Sourcing Concepts 2–19
I/O “Common” Terminal Concepts 2–20
Connecting DC I/O to “Solid State” Field Devices 2–21
Solid State Input Sensors 2–21
Solid State Output Loads 2–21
Relay Output Guidelines 2–23
Transient Suppression for Inductive Loads in a Control System 2–23
Prolonging Relay Contact Life 2–28
I/O Module Wiring and Specifications 2–30
Module Placement 2–30
I/O Module Status Indicators 2–30
Color Coding of I/O Modules 2–30
Wiring a Module with a Terminal Block 2–31
Wiring 32 and 64 Point I/O Modules 2–32
Part Numbers for Module Connectors 2–32
Ribbon Cable 2–33
Ribbon Cable Connectors 2–33
Interface Terminal Block 2–33
I/O Wiring Checklist 2–34
DL405 Input Module Chart 2–35
DL405 Output Module Chart 2–35
Begin Input Module Specifications 2–36
Begin Output Module Specifications 2–42
Glossary of Specification Terms 2–50

Chapter 3: CPU Specifications and Operation

Overview 3–2
General CPU Features 3–2
DL430 CPU Features 3–2
DL440 CPU Features 3–2
DL450 CPU Features 3–3
CPU General Specifications 3–4
CPU Electrical Specifications 3–5
CPU Hardware Features 3–6
Keyswitch Functions 3–7
Status Indicators 3–7

Setting the CPU DIP Switches 3–8
Communication Ports 3–9
Port 0 Specifications 3–9
Port 1 Specifications 3–9
Port 2 Specifications 3–10
Port 3 Specifications 3–11
Using Battery Backup 3–12
Selecting the Program Storage Media 3–13
Volatile and Non-volatile Memory 3–13
Memory Storage Types 3–13
Memory Cartridge 3–14
Memory Cartridge Capacity Table 3–14
CPU Setup 3–15
Setting the Clock and Calendar 3–15
Variable/Fixed Scan Time Feature 3–15
Password Protection 3–15
Auxiliary Functions 3–16
Clearing an Existing Program 3–17
Initializing System Memory 3–17
Setting the CPU Network Address 3–17
Setting Retentive Memory Ranges 3–17
CPU Operation 3–18
CPU Operating System 3–18
Program Mode Operation 3–19
Run Mode Operation 3–19
Read Inputs 3–20
Read Inputs from Specialty and Remote I/O 3–20
Service Peripherals and Force I/O 3–20
Update Special Relays and Special Registers 3–21
CPU Bus Communication 3–21
Update Clock, Special Relays, and Special Registers 3–21
Solve Application Program 3–22
Solve PID Loop Equations 3–22
Write Outputs 3–22
Write Outputs to Specialty and Remote I/O 3–23
Diagnostics 3–23
I/O Response Time 3–24
Is Timing Important for Your Application? 3–24
Normal Minimum I/O Response 3–24
Normal Maximum I/O Response 3–24
Improving Response Time 3–25
CPU Scan Time Considerations 3–26
Initialization Process 3–27
Reading Inputs 3–27
Reading Inputs from Specialty I/O 3–28
Service Peripherals 3–28
CPU Bus Communication 3–29
Update Clock/Calendar, Special Relays, Special Registers 3–29
Writing Outputs 3–29
Writing Outputs to Specialty I/O 3–30
Diagnostics 3–30

Application Program Execution 3–31
PLC Numbering Systems 3–32
PLC Resources 3–32
V–Memory 3–33
Binary-Coded Decimal Numbers 3–33
Hexadecimal Numbers 3–33
Memory Map 3–34
Octal Numbering System 3–34
Discrete and Word Locations 3–34
V–Memory Locations for Discrete Memory Areas 3–34
Input Points (X Data Type) 3–35
Output Points (Y Data Type) 3–35
Control Relays (C Data Type) 3–35
Timers and Timer Status Bits (T Data type) 3–35
Timer Current Values (V Data Type) 3–36
Counters and Counter Status Bits (CT Data type) 3–36
Counter Current Values (V Data Type) 3–36
Word Memory (V Data Type) 3–36
Stages (S Data type) 3–37
Special Relays (SP Data Type) 3–37
Remote I/O Points (GX Data Type) 3–37
System Parameters (V Data Type) 3–38
DL430 Memory Map 3–40
DL440 Memory Map 3–41
DL450 Memory Map 3–42
DL405 Aliases 3–43
X Input/Y Output Bit Map 3–44
Control Relay Bit Map 3–46
Timer and Counter Status Bit Maps 3–50
Remote I/O Bit Map 3–51
Stage Control / Status Bit Map 3–55

Chapter 4: System Design and Configuration

DL405 System Design Strategies 4–2
I/O System Configurations 4–2
Networking Configurations 4–3
Module Placement and Configuration 4–4
Valid Module/Unit Locations 4–4
I/O Configuration Methods 4–5
Automatic Configuration 4–5
Manual Configuration 4–5
Removing a Manual Configuration 4–6
Power–On I/O Configuration Check 4–6
Calculating the Power Budget 4–7
Managing your Power Resource 4–7
CPU Power Specifications 4–7
Module Power Requirements 4–7
Power Budget Calculation Example 4–9

Power Budget Calculation Worksheet 4–10
Local I/O Expansion 4–11
Local Base and I/O 4–11
Local Expansion Base and I/O 4–11
Remote I/O Expansion 4–12
How to Add Remote I/O Channels 4–12
Configuring the CPU's Remote I/O Channel 4–13
Configure Remote I/O Slaves 4–15
Configuring the Remote I/O Table 4–15
Remote I/O Setup Program 4–16
Remote I/O Test Program 4–17
Network Connections to MODBUS and DirectNet 4–18
Configuring the CPU's Comm Ports 4–18
MODBUS Port Configuration 4–20
DirectNET Port Configuration 4–21
Network Slave Operation 4–22
MODBUS Function Codes Supported 4–22
MODBUS Data Types Supported 4–22
Determining the MODBUS Address 4–23
If Your Host Software Requires the Data Type and Address... 4–23
Example 1: V2100 4–24
Example 2: Y20 4–24
Example 3: T10 Current Value 4–24
Example 4: C54 4–24
If Your MODBUS Host Software Requires an Address ONLY 4–25
Example 1: V2100 584/984 Mode 4–27
Example 2: Y20 584/984 Mode 4–27
Example 3: T10 Current Value 484 Mode 4–27
Example 4: C54 584/984 Mode 4–27
Network Master Operation 4–28
Step 1: Identify Master Port # and Slave # 4–29
Step 2: Load Number of Bytes to Transfer 4–29
Step 3: Specify Master Memory Area 4–30
Step 4: Specify Slave Memory Area 4–30
Communications from a Ladder Program 4–31
Multiple Read and Write Interlocks 4–31

Chapter 5: Standard RLL Instructions

Introduction 5–2
Using Boolean Instructions 5–4
END Statement 5–4
Simple Rungs 5–4
Normally Closed Contact 5–5
Contacts in Series 5–5
Midline Outputs 5–5
Parallel Elements 5–6
Joining Series Branches in Parallel 5–6
Joining Parallel Branches in Series 5–6
Combination Networks 5–7

Comparative Boolean 5–7
Boolean Stack 5–8
Immediate Boolean 5–9
Boolean Instructions 5–10
Comparative Boolean Instructions 5–26
Immediate Instructions 5–32
Timer, Counter, and Shift Register Instructions 5–41
Using Timers 5–41
Timer (TMR) and Timer Fast (TMRF) 5–42
Timer Example Using Discrete Status Bits 5–43
Timer Example Using Comparative Contacts 5–43
Accumulating Timer (TMRA) and Accumulating Fast Timer (TMRAF) 5–44
Accumulating Timer Example using Discrete Status Bits 5–45
Accumulator Timer Example Using Comparative Contacts 5–45
Using Counters 5–46
Counter (CNT) 5–47
Counter Example Using Discrete Status Bits 5–48
Counter Example Using Comparative Contacts 5–48
Stage Counter (SGCNT) 5–49
Stage Counter Example Using Discrete Status Bits 5–50
Stage Counter Example Using Comparative Contacts 5–50
Up/Down Counter (UDC) 5–51
Up/Down Counter Example Using Discrete Status Bits 5–52
Up/Down Counter Example Using Comparative Contacts 5–52
Shift Register (SR) 5–53
Accumulator/Data Stack Load and Output Instructions 5–54
Using the Accumulator 5–54
Copying Data to the Accumulator 5–54
Changing the Accumulator Data 5–55
Using the Accumulator Stack 5–56
Accumulator and Accumulator Stack Memory Locations 5–57
Using Pointers 5–58
Accumulator Logic Instructions 5–70
Math Instructions 5–87
Transcendental Functions 5–118
Bit Operation Instructions 5–121
Number Conversion Instructions 5–128
Shuffle Digits Block Diagram 5–140
Table Instructions 5–142
Copy Data From a Data Label Area to V–memory 5–163
Copy Data From V–memory to a Data Label Area 5–165
Clock/Calender Instructions 5–175
CPU Control Instructions 5–177
Program Control Instructions 5–179
Interrupt Instructions 5–185
Intelligent I/O Instructions 5–189
Network Instructions 5–191
Message Instructions 5–194

Chapter 6: Drum Instruction Programming (DL450 CPU only)

Introduction **6-2**
Purpose 6-2
Drum Terminology 6-2
Drum Chart Representation 6-3
Output Sequences 6-3
Step Transitions **6-4**
Drum Instruction Types 6-4
Timer-Only Transitions 6-4
Timer and Event Transitions 6-5
Event-Only Transitions 6-6
Counter Assignments 6-6
Last Step Completion 6-7
Overview of Drum Operation **6-8**
Drum Instruction Block Diagram 6-8
Powerup State of Drum Registers 6-9
Drum Control Techniques **6-10**
Drum Control Inputs 6-10
Self-Resetting Drum 6-11
Initializing Drum Outputs 6-11
Cascaded Drums Provide More Than 16 Steps 6-12
Handheld Programmer Drum Mnemonics 6-13
Drum Instructions **6-15**
Timed Drum with Discrete Outputs (DRUM) 6-15
Event Drum with Discrete Outputs (EDRUM) 6-18
Masked Event Drum with Discrete Outputs (MDRMD) 6-21
Masked Event Drum with Word Output (MDRMW) 6-24

Chapter 7: RLL^{PLUS} Stage Programming

Introduction to Stage Programming **7-2**
Overcoming “Stage Fright” 7-2
Learning to Draw State Transition Diagrams **7-3**
Introduction to Process States 7-3
The Need for State Diagrams 7-3
A 2-State Process 7-3
RLL Equivalent 7-4
Stage Equivalent 7-4
Let’s Compare 7-5
Initial Stages 7-5
What Stage Bits Do 7-6
Stage Instruction Characteristics 7-6
Using the Stage Jump Instruction for State Transitions **7-7**
Stage Jump, Set, and Reset Instructions 7-7
Stage Program Example: Toggle On/Off Lamp Controller **7-8**
A 4-State Process 7-8
Four Steps to Writing a Stage Program **7-9**
Stage Program Example: A Garage Door Opener **7-10**

Garage Door Opener Example 7–10
Draw the Block Diagram 7–10
Draw the State Diagram 7–11
Add Safety Light Feature 7–12
Modify the Block Diagram and State Diagram 7–12
Using a Timer Inside a Stage 7–13
Add Emergency Stop Feature 7–14
Exclusive Transitions 7–14
Stage Program Design Considerations 7–15
Stage Program Organization 7–15
How Instructions Work Inside Stages 7–16
Using a Stage as a Supervisory Process 7–17
Stage Counter 7–17
Unconditional Outputs 7–18
Power Flow Transition Technique 7–18
Parallel Processing Concepts 7–19
Parallel Processes 7–19
Converging Processes 7–19
Convergence Stages (CV) 7–19
Convergence Jump (CVJMP) 7–20
Convergence Stage Guidelines 7–20
Managing Large Programs 7–21
Stage Blocks (BLK, BEND) 7–21
Block Call (BCALL) 7–22
RLL^{PLUS} Instructions 7–23
Stage (SG) 7–23
Initial Stage (ISG) 7–24
Jump (JMP) 7–24
Not Jump (NJMP) 7–24
Converge Stage (CV) and Converge Jump (CVJMP) 7–25
Block Call (BCALL) 7–27
Block (BLK) 7–27
Block End (BEND) 7–27
Questions and Answers about Stage Programming 7–29

Chapter 8: PID Loop Operation (DL450 only)

DL450 PID Control 8–2
DL450 PID Control Features 8–2
Introduction to PID Control 8–4
What is PID Control? 8–4
Introducing DL450 PID Control 8–6
Process Control Definitions 8–8
PID Loop Operation 8–9
PID Position Algorithm 8–9
Reset Windup Protection 8–10
Freeze Bias 8–11
Adjusting the Bias 8–11
Step Bias Proportional to Step Change SP 8–12
Eliminating Proportional, Integral or Derivative Action 8–12

Velocity Form of the PID Equation 8–12
Bumpless Transfer 8–13
Loop Alarms 8–13
Loop Operating Modes 8–14
Special Loop Calculations 8–14
Ten Steps to Successful Process Control 8–16
PID Loop Setup 8–18
Some Things to Do and Know Before Starting 8–18
PID Error Flags 8–18
Establishing the Loop Table Size and Location 8–19
Loop Table Word Definitions 8–21
PID Mode Setting 1 Bit Descriptions (Addr + 00) 8–22
PID Mode Setting 2 Bit Descriptions (Addr + 01) 8–23
Mode/Alarm Monitoring Word (Addr + 06) 8–24
Ramp/Soak Table Flags (Addr + 33) 8–24
Ramp/Soak Table Location (Addr + 34) 8–25
Ramp/Soak Table Programming Error Flags (Addr + 35) 8–25
Configure the PID Loop 8–26
PID Loop Tuning 8–40
Open–Loop Test 8–40
Manual Tuning Procedure 8–41
Alternative Manual Tuning Procedures by Others 8–44
Tuning PID Controllers 8–44
Auto Tuning Procedure 8–45
Use DirectSOFT 5 Data View with PID View 8–49
Open a New Data View Window 8–49
Open PID View 8–49
Using the Special PID Feature 8–52
How to Change Loop Modes 8–52
Operator Panel Control of PID Modes 8–53
PLC Modes' Effect on Loop Modes 8–53
Loop Mode Override 8–53
Creating an Analog Filter in Ladder Logic 8–54
Use the DirectSOFT 5 Filter Intelligent Box Instruction 8–55
FilterB Example 8–55
Ramp/Soak Generator 8–56
Introduction 8–56
Ramp/Soak Table 8–57
Ramp/Soak Table Flags 8–59
Ramp/Soak Generator Enable 8–59
Ramp/Soak Controls 8–59
Ramp/Soak Profile Monitoring 8–60
Ramp/Soak Programming Errors 8–60
Testing Your Ramp/Soak Profile 8–60
DirectSOFT Ramp/Soak Example 8–61
Setup the Profile in PID Setup 8–61
Program the Ramp/Soak Control in Relay Ladder 8–61
Test the Profile 8–62
Cascade Control 8–63
Introduction 8–63
Cascaded Loops in the DL450 CPU 8–64

Tuning Cascaded Loops 8–65
Time-Proportioning Control 8–66
On/Off Control Program Example 8–67
Feedforward Control 8–68
Feedforward Example 8–69
PID Example Program 8–70
Program Setup for the PID Loop 8–70
Troubleshooting Tips 8–72
Glossary of PID Loop Terminology 8–74
Bibliography 8–76

Chapter 9: Maintenance and Troubleshooting

Hardware Maintenance 9–2
Standard Maintenance 9–2
CPU Battery Replacement 9–2
CMOS RAM Memory Cartridge Battery Replacement 9–3
Diagnostics 9–4
Diagnostics 9–4
Fatal Errors 9–4
Non-fatal Errors 9–4
Finding Diagnostic Information 9–4
V-memory Error Code Locations 9–5
Special Relays (SP) Corresponding to Error Codes 9–5
I/O Module Codes 9–6
Error Message Tables 9–7
System Error Codes 9–8
CPU Status Indicators 9–9
PWR Indicator 9–10
Run Indicator 9–11
CPU Indicator 9–12
BATT Indicator 9–12
DIAG Indicator 9–12
I/O Indicator 9–12
COM Indicator 9–13
TXD and RXD Indicators 9–13
I/O Module Troubleshooting 9–14
Some Quick Steps 9–14
Testing Output Points 9–15
Handheld Programmer Keystrokes Used to Test an Output Point 9–15
Noise Troubleshooting 9–16
Electrical Noise Problems 9–16
Reducing Electrical Noise 9–16
Machine Startup and Program Troubleshooting 9–17
Syntax Check 9–17
Duplicate Reference Check 9–18
TEST-PGM and TEST-RUN Modes 9–19
Run Time Edits 9–21
Changing an Instruction During Run Mode 9–22

Inserting an Instruction During Run Mode 9–23
Deleting an Instruction During Run Mode 9–24
Special Debug Instructions 9–25

Appendix A: Auxiliary Functions

Introduction A–2
What are Auxiliary Functions? A–2
Accessing AUX Functions via ***Direct***SOFT A–3
Accessing AUX Functions via the Handheld Programmer A–3
AUX 1* — Operating Modes A–4
AUX 11 Go to Run Mode A–4
AUX 12 Go to Test Mode A–4
AUX 13 Go to Program Mode A–4
AUX 14 Run Time Edit A–4
AUX 2* — RLL Operations A–5
AUX 21 Check Program A–5
AUX 22 Change Reference A–5
AUX 23 Clear Ladder Range A–5
AUX 24 Clear Ladders A–5
AUX 25 Select MC or Flash Memory A–5
AUX 26 Copy MC Contents to Flash A–5
AUX 27 Copy Flash Contents to MC A–5
AUX 28 Verify Flash Contents = MC A–5
AUX 3* — V-memory Operations A–6
AUX 31 Clear V–Memory A–6
AUX 32 Clear V-memory Range A–6
AUX 33 Find V-memory Value A–6
AUX 4* — I/O Configuration A–6
AUX 41 Show I/O Configuration A–6
AUX 42 I/O Diagnostics A–6
AUX 44 Power-up Configuration Check A–6
AUX 45 Select Configuration A–7
AUX 46 I/O Configuration A–7
AUX 47 Intelligent I/O Monitor A–7
AUX 5* — CPU Configuration A–8
AUX 51 Modify Program Name A–8
AUX 52 Display /Change Calendar A–8
AUX 53 Display Scan Time A–8
AUX 54 Initialize Scratchpad A–8
AUX 55 Set Watchdog Timer A–8
AUX 56 CPU Network Address A–9
AUX 57 Set Retentive Ranges A–9
AUX 58 Test Operations A–9
AUX 5C Display Error History A–9
AUX 5D Select PLC Scan Mode A–10
AUX 6* — Handheld Programmer Configuration A–10
AUX 61 Show Revision Numbers A–10
AUX 62 Beeper On / Off A–10
AUX 63 Turning Off the Backlight A–10

AUX 64 Select Online / Offline A–10
AUX 65 Run Self Diagnostics A–10
AUX 7* — Memory Cartridge Operations A–11
Transferrable Memory Areas A–11
AUX 71 CPU to Memory Cartridge A–11
AUX 72 Memory Cartridge to CPU A–11
AUX 73 Compare Memory Cartridge to CPU A–11
AUX 74 Memory Cartridge Blank Check A–11
AUX 75 Clear Memory Cartridge A–11
AUX 76 Display Memory Cartridge Type A–11
AUX 77 Tape to Memory Cartridge A–11
AUX 78 Memory Cartridge to Tape A–12
AUX 79 Compare Memory Cartridge to Tape A–12
AUX 8* — Password Operations A–12
AUX 81 Modify Password A–12
AUX 82 Unlock CPU A–13
AUX 83 Lock CPU A–13

Appendix B: DL405 Error Codes

Appendix C: Instruction Execution Times

Introduction C–2
V-Memory Data Registers C–2
V-Memory Bit Registers C–2
How to Read the Tables C–2
Boolean Instructions C–3
Comparative Boolean Instructions C–5
Immediate Instructions C–11
Clock/Calendar Instructions C–11
Timer, Counter, and Shift Register Instructions C–12
Accumulator/Data Stack Load and Output Instructions C–13
Accumulator Logic Instructions C–14
Bit Operation Instructions C–15
Math Instructions C–16
Number Conversion Instructions C–19
Table Instructions C–20
CPU Control Instructions C–20
Program Control Instructions C–21
Interrupt Instructions C–21
RLL^{PLUS} Instructions C–21
Intelligent I/O Instructions C–22
Network Instructions C–22
Message Instructions C–22
Drum Instructions C–22

Appendix D: Special Relays

Startup and Real-time Special Relays **D–2**
CPU Status Relays **D–2**
System Monitoring Relays **D–3**
Accumulator Status Relays **D–3**
Communication Monitoring Relays **D–4**

Appendix E: DL405 Product Weights

Product Weight Table **E–2**

Appendix F: PLC Memory

DL405 PLC Memory **F–2**

Appendix G: ASCII Table

ASCII Conversion Table **G–2**

Appendix H: Numbering Systems

Introduction **H–2**
Binary Numbering System **H–2**
Hexadecimal Numbering System **H–3**
Octal Numbering System **H–4**
Binary Coded Decimal (BCD) Numbering System **H–5**
Real (Floating Point) Numbering System **H–6**
BCD/Binary/Decimal/Hex/Octal – What is the Difference? **H–7**
Data Type Mismatch **H–8**
Signed vs. Unsigned Integers **H–9**
AutomationDirect.com Products and Data Types **H–10**
DirectLOGIC PLCs H–10
C–more/C–more Micro–Graphic Panels H–10

Appendix I: European Union Directives (CE)

European Union (EU) Directives **I–2**
Member Countries I–2
General Safety I–4
Special Installation Manual I–4
Other Sources of Information I–4
Basic EMC Installation Guidelines **I–5**
Enclosures I–5
Suppression and Fusing I–5

Internal Enclosure Grounding I–6
Equi–potential Grounding I–6
Communications and Shielded Cables I–6
Analog and RS232 Cables I–7
Multidrop Cables I–7
Shielded Cables within Enclosures I–7
Analog Modules and RF Interference I–8
Network Isolation I–8
DC Powered Versions I–8
Items Specific to the DL405 . . . . . I–8

Index

1

Getting Started

In This Chapter. . . .

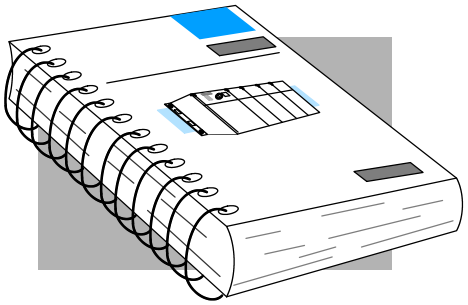
- Introduction
- DL405 System Components
- Programming Methods
- ***Direct***LOGIC™ Part Numbering System
- Quick Start for PLC Checkout and Programming
- Steps to Designing a Successful System
- Frequently Asked Questions

Introduction

The Purpose of this Manual

Thank you for purchasing our DL405 family of products. This manual shows you how to install, program, and maintain the equipment. It also helps you understand how to interface them to other devices in a control system.

This manual contains important information for personnel who will install DL405 PLCs and components, and for the PLC programmer. If you understand PLC systems our manuals will provide all 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 a new DL405 customer, we suggest you read this manual completely so you can understand the wide variety of features in the DL405 family of products. We believe you will be pleasantly surprised with how much you can accomplish with our products.

Supplemental Manuals

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

Technical Support

We realize that even though we strive to be the best, we may have arranged our information in such a way you cannot find what you are looking for. First, check these resources for help in locating the information:

- **Table of Contents** – chapter and section listing of contents, in the front of this manual
- **Appendices** – reference material for key topics, near the end of this manual
- **Index** – alphabetical listing of key words, at the end of this manual

You can also check our online resources for the latest product support information:

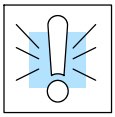
- **Internet** – Our Web address is **http://www.automationdirect.com**

If you still need assistance, please call us at 770-844-4200. Our technical support group is glad to work with you in answering your questions. They are available Monday through Friday from 9:00 A.M. to 6:00 P.M. Eastern Standard Time. If you have a comment or question about any of our products, services, or manuals, please fill out and return the ‘Suggestions’ card that was shipped with this manual.

Conventions Used

When you see the “light bulb” icon in the left-hand margin, the paragraph to its immediate right will give you a **special tip**.
The word **TIP:** in boldface will mark the beginning of the text.

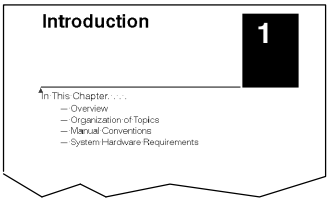
When you see the “notepad” icon in the left-hand margin, the paragraph to its immediate right will be a **special note**.
The word **NOTE:** in boldface will mark the beginning of the text.



When you see the “exclamation mark” icon in the left-hand margin, the paragraph to its immediate right will be a **warning**. This information could prevent injury, loss of property, or even death (in extreme cases).
The word **WARNING:** and text will be in **boldface**.

Key Topics for Each Chapter

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



DL405 System Components

The DL405 product family is one of the most versatile and widely accepted PLCs used for medium control applications. The CPUs are small, yet powerful. Their modular design and expansion capability blend well with todays fast-moving industry. The following is a summary of the major DL405 system components.

CPUs

There are three feature-enhanced CPUs in this product line, the DL430, DL440, and the DL450. All 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 DL450 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 DL430/440 CPUs can support up to 640 I/O between the CPU base and expansion bases (the DL450 up to 1024 local I/O). A maximum of 512 additional I/O for the DL430, 1024 for the DL440, or 1536 for the DL450 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 some of the most powerful modules in the industry. They include 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 a 100KHz high-speed input, thermocouple, general purpose communication, magnetic pulse input, 16 loop PID function and more.

Programming Methods

There are two programming methods available to the DL405 CPUs, RLL (Relay Ladder Logic) and RLL*PLUS*.Stage Programing. Both the ***Direct***SOFT programming package and the handheld programmer support RLL and Stage.

***Direct*SOFT Programming for Windows™**

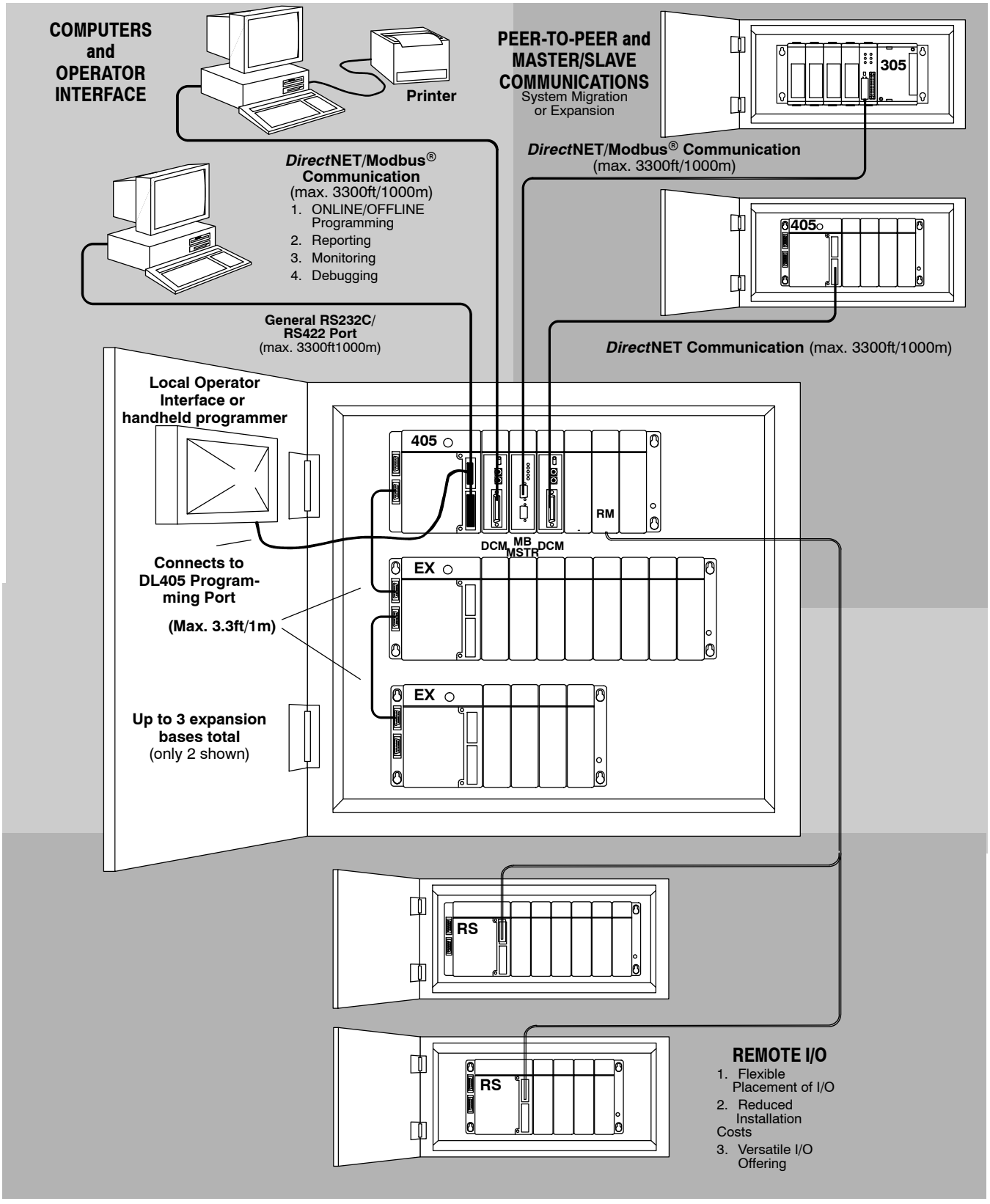
The DL405 CPUs can be programmed with one of the most advanced programming packages in the industry --***Direct***SOFT 5, Ver. 5 or later. ***Direct***SOFT 5 is a Windows-based software package that supports many of the Windows features you already know, such as cut-and-paste between applications, point-and-click editing, viewing and editing multiple application programs at the same time, browsers, and newly added Intelligent Box (IBox) Instructions which will help ease your programming tasks. ***Direct***SOFT 5 universally supports the ***Direct***LOGIC™ CPU families. This means you can use the same ***Direct***SOFT 5 package to program DL105, DL205, DL305, DL405 or any new CPUs that we add to our product line. There is a separate manual for the ***Direct***SOFT 5 programming software.

Handheld Programmer

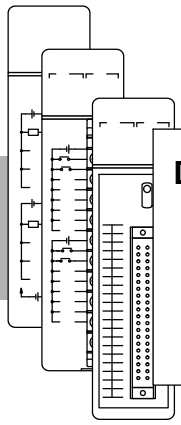
All DL405 CPUs have a built-in programming port for use with the handheld programmer (D4-HPP). The HPP can be used to create, modify and debug your application program. A separate manual for the DL405 HPP is available.

DL405 System Diagrams

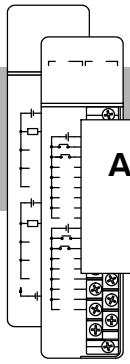
The diagram below shows the major components and configurations of the DL405 system. The next two pages show specific components for building your system.



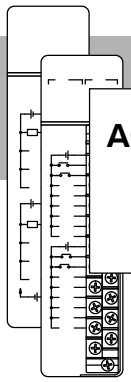
DirectLOGIC™



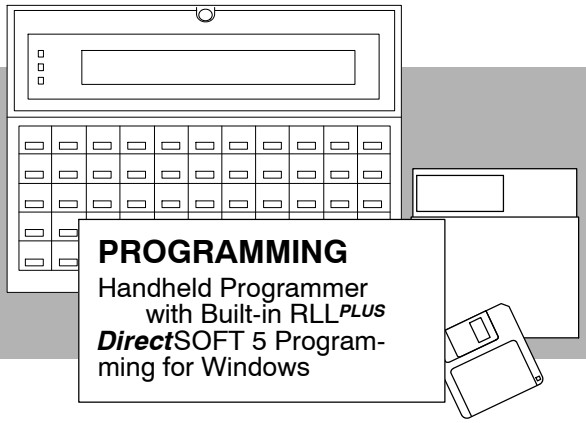
DC INPUT
8pt 24–48 VDC
16pt 12–24 VDC
16pt 12–24 VDC (1ms response)
32pt 24 VDC
32pt 5–12 VDC
64pt 24VDC



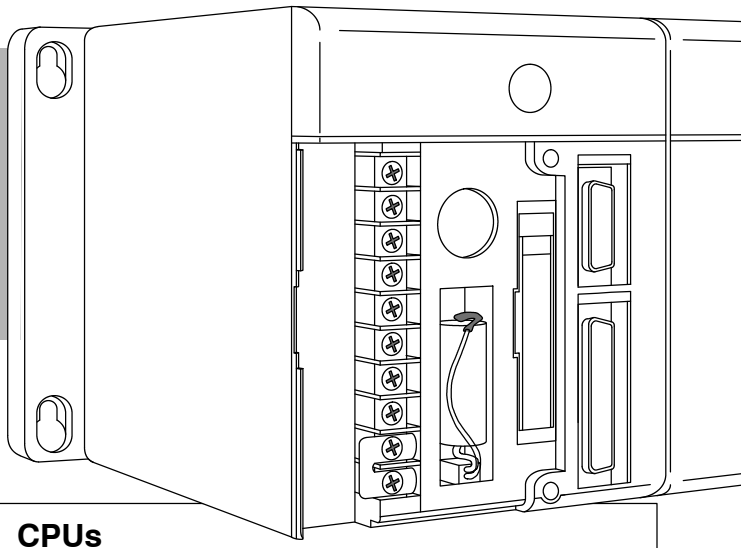
AC INPUT
8pt 110–200 VAC
16pt 110 VAC

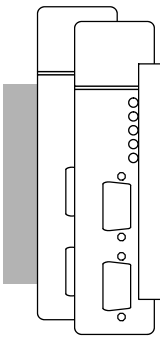


AC/DC INPUT
8pt 90–150 VAC/DC Isolated
16pt 12–24 VAC/DC

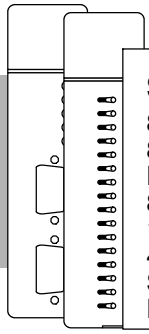


PROGRAMMING
Handheld Programmer with Built-in RLL*PLUS*
DirectSOFT 5 Programming for Windows





COPROCESSOR™ MODULES
RS232C / RS422 / RS485
Telephone Modem
Program Memory Ranges from 128K to 512K



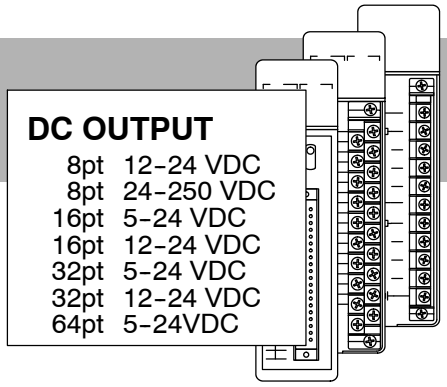
SPECIALTY MODULES
8/16pt Input Simulator
8pt Interrupt Input Module
High Speed Counter Module
8 pt. Magnetic Pulse Input Module
16 Loop PID Module
4 Loop Temperature Controller
SDS™ Interface Module
Filler Module

CPUs
DL430 – 110/220 VAC P/S,
6.5K Built-in EPROM Memory
DL440 – 110/220 VAC P/S,
22.5K Memory (Memory Cartridge Required)
DL440 – 24 VDC or 125 VDC P/S,
22.5K Memory (Memory Cartridge Required)
DL450 – 110/220 VAC P/S
7.5K Built-in Flash Memory
DL450 – 24 VDC or 125 VDC P/S
7.5K Built-in Flash Memory

BASES
4 Slot Bases (expandable or nonexpandable)
6 Slot Bases (expandable or nonexpandable)
8 Slot Base (expandable or nonexpandable)

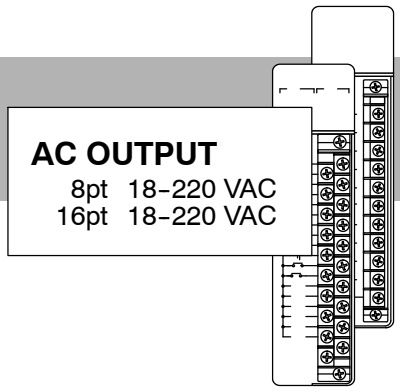
MEMORY CARTRIDGES
CMOS RAM w/battery
UVPROM
EEPROM

DL405 Family



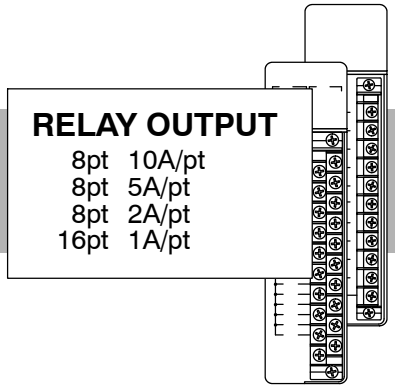
DC OUTPUT

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



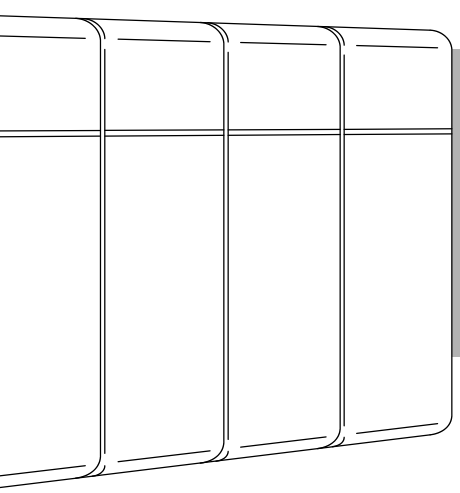
AC OUTPUT

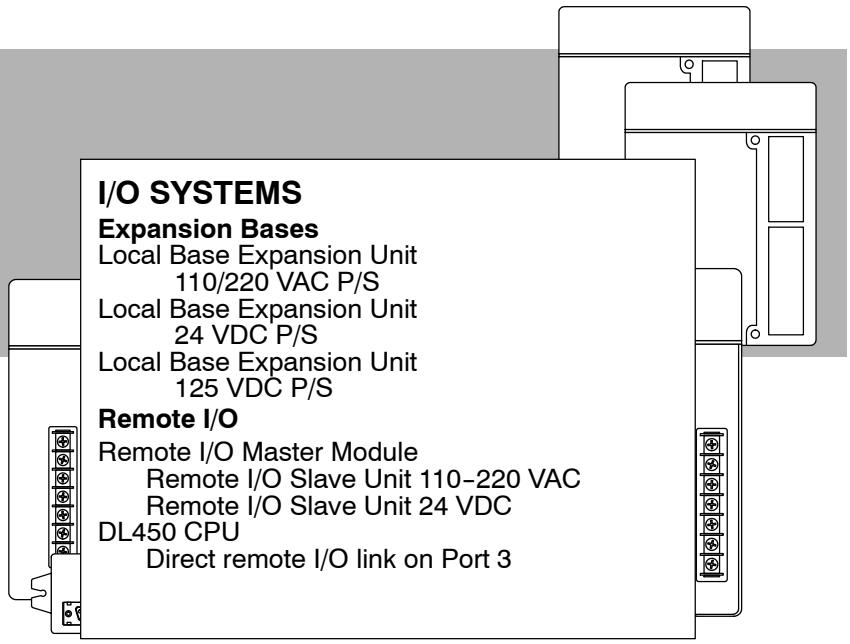
8pt 18–220 VAC
16pt 18–220 VAC



RELAY OUTPUT

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





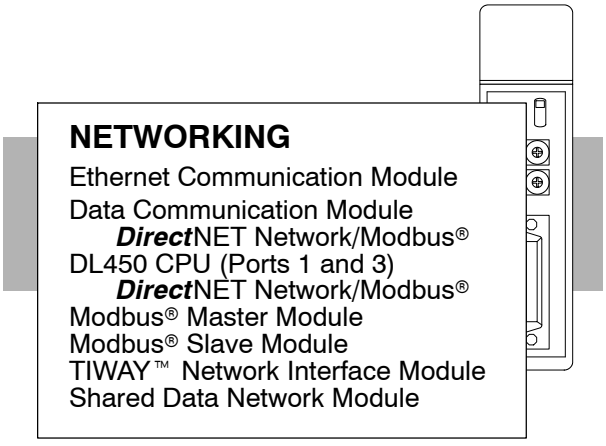
I/O SYSTEMS

Expansion Bases

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

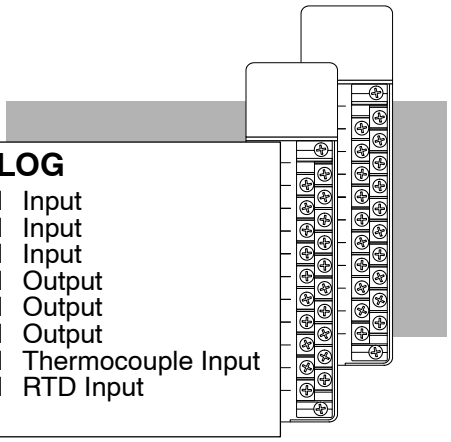
Remote I/O

Remote I/O Master Module
Remote I/O Slave Unit 110–220 VAC
Remote I/O Slave Unit 24 VDC
DL450 CPU
Direct remote I/O link on Port 3



NETWORKING

Ethernet Communication Module
Data Communication Module
DirectNET Network/Modbus®
DL450 CPU (Ports 1 and 3)
DirectNET Network/Modbus®
Modbus® Master Module
Modbus® Slave Module
TIWAY™ Network Interface Module
Shared Data Network Module



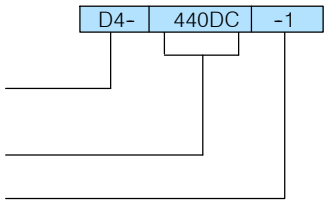
ANALOG

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

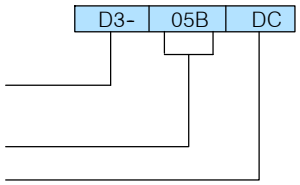
*Direct*LOGIC™ Part Numbering System

A DL405 PLC control system may be comprised of many products from the DL405 family. The part numbering system below shows how the part numbering systems work for each product category. Part numbers for accessory items such as cables, batteries, memory cartridges etc. are typically an abbreviation of the description for the item.

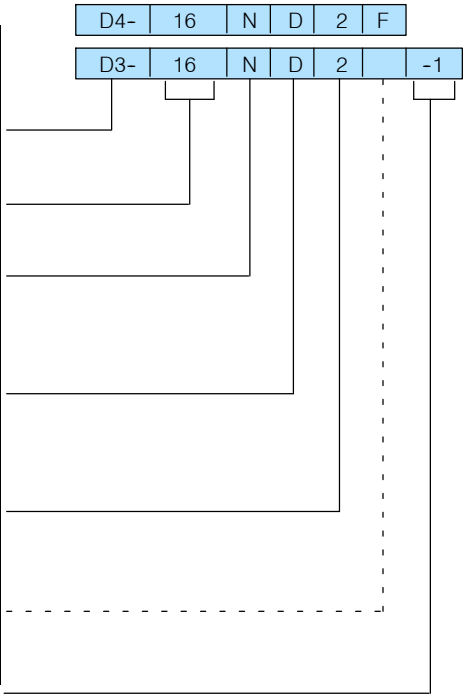
CPUs and Micro PLCs **Specialty CPUs**	
Product family	D1/F1 D2/F2 D3/F3 D4/F4
Class of CPU / Abbreviation	230...,330...,430...
Denotes a differentiation between similar modules	-1, -2, -3, -4



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

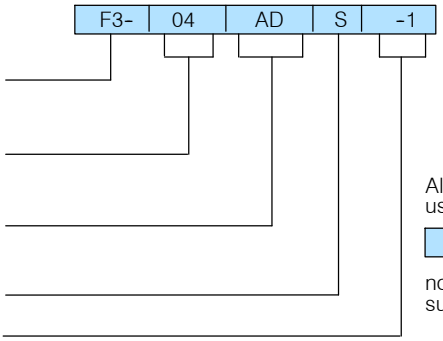


Discrete I/O	
DL205 Product family DL305 Product family DL405 Product family	D2/F2 D3/F3 D4/F4
Number of points	04/08/12/16/32/64
Input Output Combination	N T C
AC DC Either Relay	A D E R
Current Sinking Current Sourcing Current Sinking/Sourcing	1 2 3
High Current Isolation Fast I/O	H S F
Denotes a differentiation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



*Direct*LOGIC™ Part Numbering System

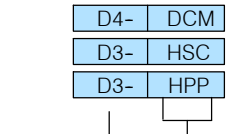
Analog I/O	
DL205 Product family	D2/F2
DL305 Product family	D3/F3
DL405 Product family	D4/F4
Number of channels	02/04/08/16
Input (Analog to Digital)	AD
Output (Digital to Analog)	DA
Combination	xADxDA
Isolated	S
Denotes a differentiation between Similar modules	-1, -2, -3, -4



Alternate example of Analog I/O using abbreviations

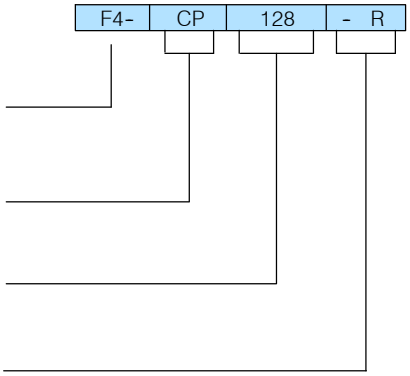
note: -n indicates thermocouple type such as: J, K, T, R, S or E

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



DCM (Data Communication Module)
HSC (High Speed Counter)
HPP (RLL PLUS Handheld Programmer)

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



Quick Start for PLC 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.

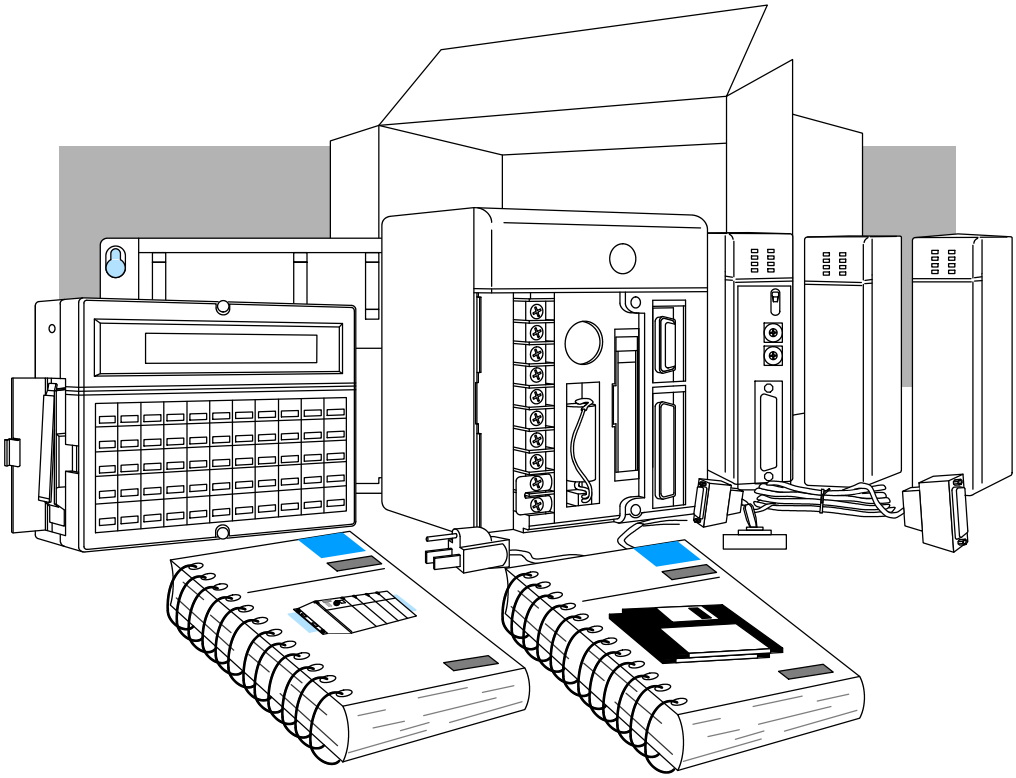
Step 1: Unpack the DL405 Equipment

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 (with a memory cartridge if you are using a DL440 or DL450)
- D4-16ND2 DC input module or a D4-16SIM input simulator module
- D4-16TR Relay output module
- Power cord
- Hook up wire
- A 24 VDC toggle switch (if not using the input simulator module)
- A screwdriver, regular or Philips type

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

- ***Direct***SOFT Programming Software, ***Direct***SOFT Manual, and a programming cable (connects the CPU to a personal computer), or
- D4-HPP Handheld Programmer (programming cable is optional), and the Handheld Programmer Manual

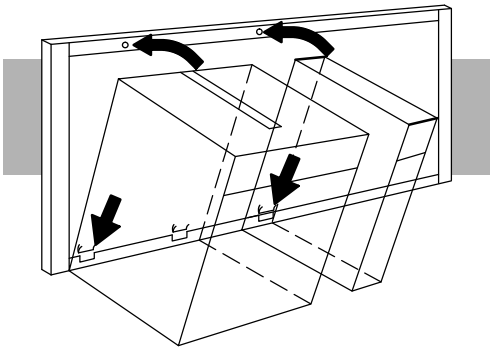


Step 2: Install the CPU and I/O Modules

Insert the CPU and I/O into the base. The CPU must 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.

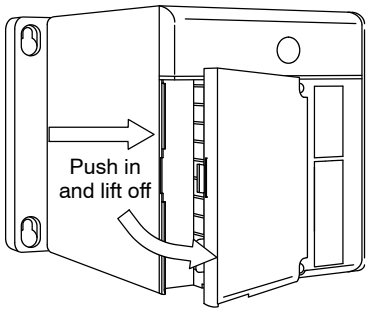
Placement of discrete, analog and relay modules are not critical and may go in any slot in any base however for this example install the output module in the slot next to the CPU (slot 0) and the input module in the next slot (slot 1). Limiting factors for other types of modules are discussed in Chapter 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 a 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.



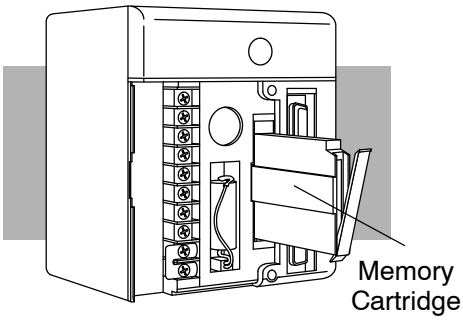
Step 3: Remove Terminal Strip Access Cover

Remove the terminal strip cover. It has a small retention snap on the left edge. Push in and up then pull the cover off.



Step 4: Install Memory Cartridge

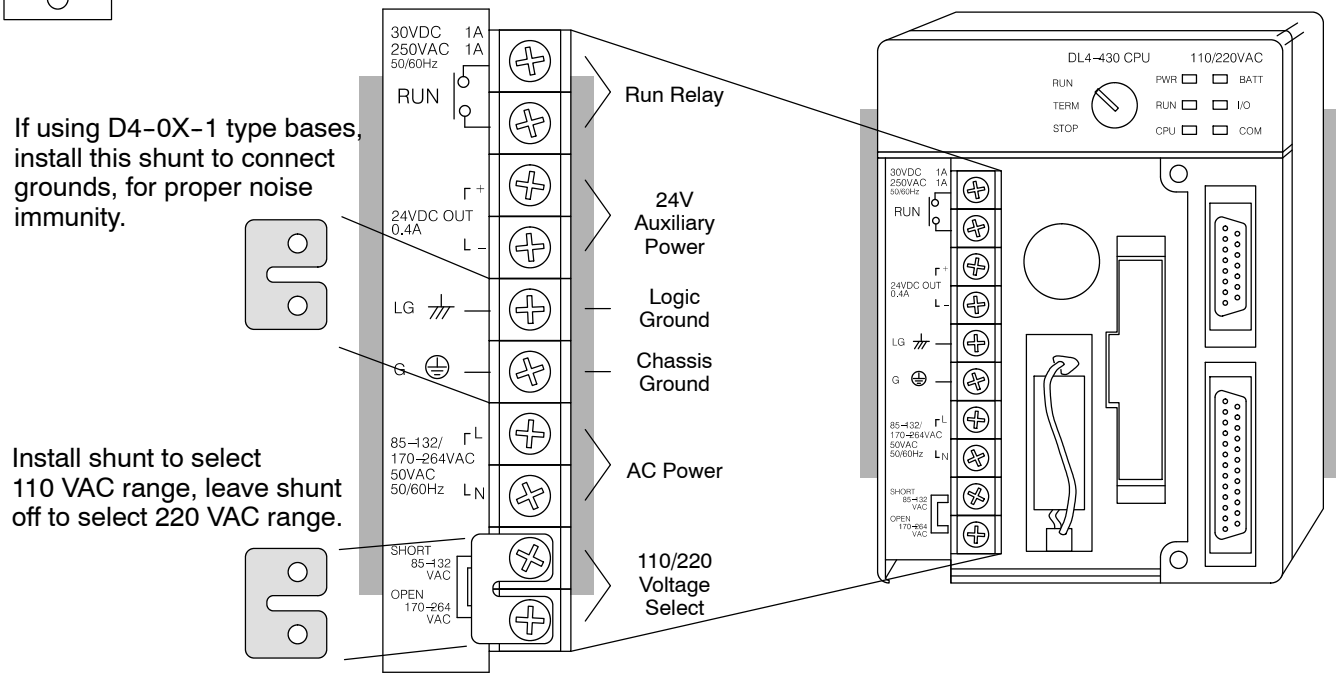
If you are using a DL440 CPU (optional on DL450) you need to install the memory cartridge in the slot to the right of the battery. Make sure it is firmly seated. To find out more about memory cartridges see Chapter 3.



Step 5: Select Operating Power Range

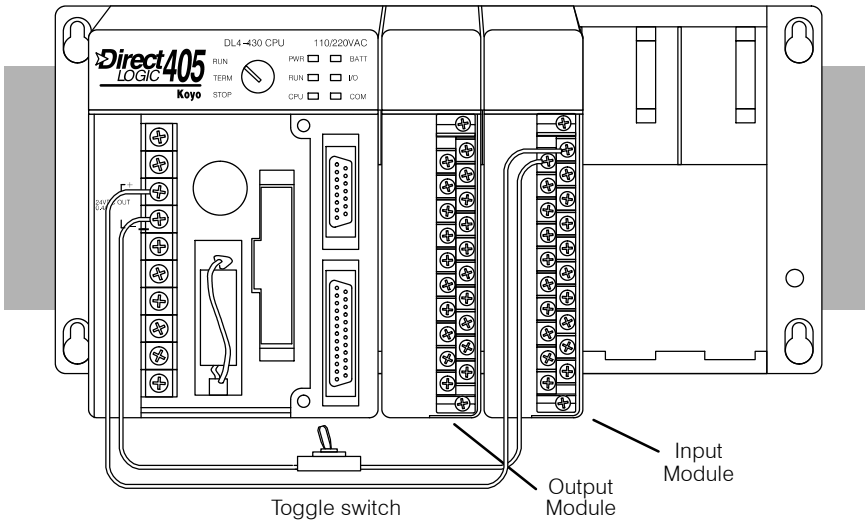
If you are using 110VAC install the voltage select jumper on the bottom two terminals. If you are using 220VAC power, do not install the jumper. You can find a detailed explanation of the terminal block on both the CPU and expansion units in Chapter 2, Installation, Wiring, and Specifications.

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



Step 6: Add I/O Simulation

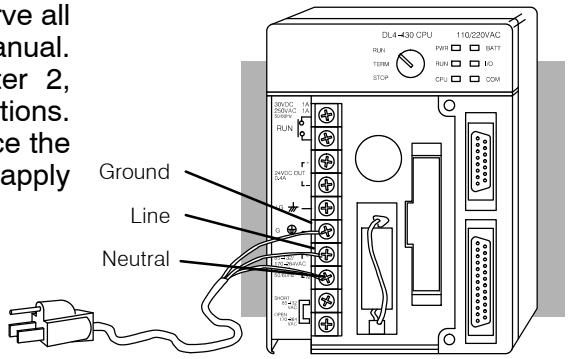
To finish this quick-start exercise or study other examples in this manual, you'll need to install an input simulator module (or wire an input switch as shown below), and add an output module. Using an input simulator is the quickest way to get physical inputs for checking out the system or a new program. 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 module (X0) to the toggle switch and 24VDC auxiliary power supply on the CPU terminal strip as shown below for the 16ND2 input module. Chapter 2, Installation, Wiring, and Specifications provides a list of I/O wiring guidelines.

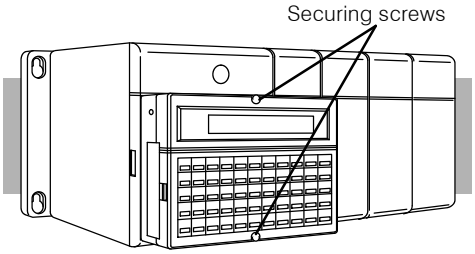
Step 7: Connect the Power Wiring

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



Step 8: Connect the Handheld Programmer

Remove the connector cover at the top right of the CPU. Join the Handheld programmer's 15 pin D type connector (located on the back) to the connector you just uncovered on the CPU. Finger tighten the securing screws on the top and bottom of the Handheld programmer.



Step 9: Switch On the System Power

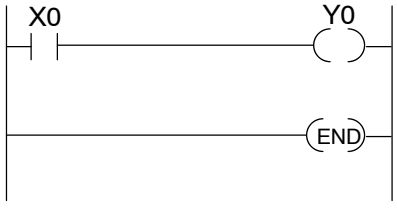
Apply power to the system and ensure the PWR indicator on the CPU is on. If not, remove power from the system and check all wiring and refer to the troubleshooting section in Chapter 9 for assistance.

Step 10: Enter the Program

Turn the key switch on the CPU to the STOP position and then back to the TERM position. This puts the CPU in the program mode and allows access to the CPU program. The PGM indicator should be illuminated on the HPP. Enter the following keystrokes on the HPP:

NOTE: It is not necessary for you to configure the I/O for this system since the DL430 CPUs automatically configure I/O and the DL440 and DL450 CPUs default to automatic I/O configuration.

CLR	CLR				
AUX	2	4	↵	↵	↵
CLR	$(AD)	0	NXT		
STR	X(IN)	0	↵		
OUT	Y(OUT)	0	↵		
END	↵				



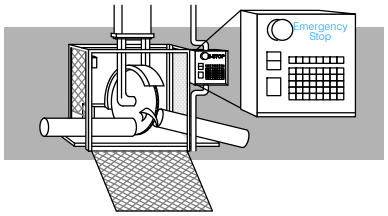
After entering the simple example program turn the key switch from the TERM position to the RUN position and back to TERM. The RUN indicator on the CPU will come on indicating the CPU has entered the run mode. If not repeat Step 10 insuring the program is entered properly or refer to the troubleshooting guide in chapter 9.

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

Steps to Designing a Successful System

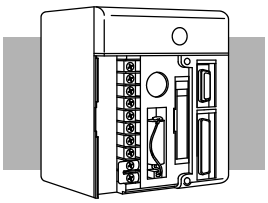
Step 1: Review the Installation Guidelines

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



Step 2: Understand the CPU Setup Procedures

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

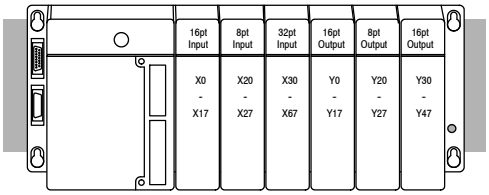


Step 3: Understand the I/O System Configurations

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

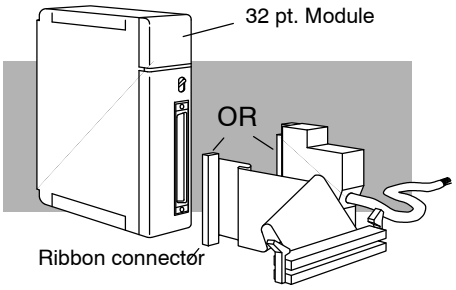
- Local System
- Expansion System
- Remote I/O System
- Network connections

It is also important to understand how the system Power Budget is calculated. See Chapter 4 for more information.



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

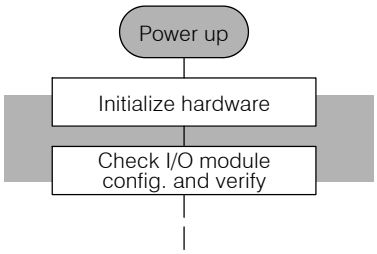
There are many different I/O modules available with the DL405 system. Chapter 2 provides the specifications and wiring diagrams for the discrete I/O modules.



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

Step 5: Understand the CPU Operation

Before you begin to enter a program, it is very helpful to understand how the DL405 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.

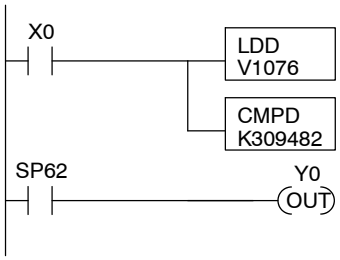


Step 6: Review the Programming Concepts

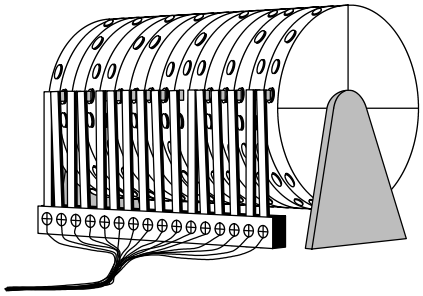
The DL405 PLC provides four main approaches to solving the application program, including the PID loop task depicted in the next figure.

- 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 DL450 has four timer/event drum types, each with up to 16 steps They offer both time and/or event-based step transitions. Drums are best for a repetitive process based on a single series of steps.
- Stage programming (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 DL450 PID Loop Operation uses setup tables to configure up to 16 loops. Features include alarms, SP ramp/soak generation, and more.

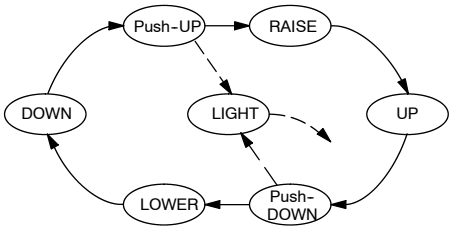
Standard RLL Programming
(see Chapter 5)



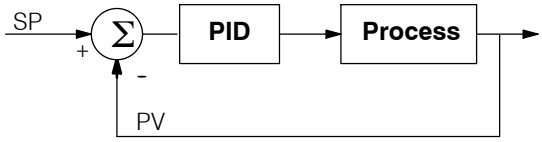
Timer/Event Drum Sequencer
(see Chapter 6)



Stage Programming
(see Chapter 7)

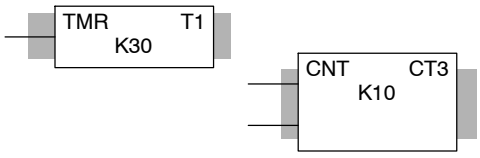


PID Loop Operation
(see Chapter 8)



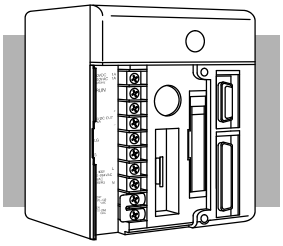
Step 7: Choose the Instructions

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!



Step 8: Understand 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 DL405 CPU back to factory defaults

A. Resetting the processor to factory defaults is a two step process. First clear the processor memory using ***Direct***SOFT **PLC>CLEAR PLC MEMORY**. 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, communications port settings, etc.

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

A. All of the 405 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 DL450?

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-ECOM module is needed to support ethernet.

Q. Can the DL405 use the Modbus protocol?

A. The DL450 supports Modbus on port 1 or 3 as a master or slave. The DL-430/440 can support Modbus using the F4-MAS-MB master or F4-SLV-MB slave module.

Q. Can I have more than 16 PID loops on a system using the DL450?

A. Yes. You can still use the 16-loop PID module in the base, plus the DL450's loops.

Q. What are the networking capabilities of the DL450?

A. The DL430/DL440 CPUs can serve as ***Direct***NET slaves. The DL450 has two network ports, which can serve as ***Direct***NET masters or slaves, or MODBUS masters or slaves.

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

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

2

Installation, Wiring, and Specifications

In This Chapter. . . .

— Safety Guidelines
— Mounting Guidelines
— Installing DL405 Bases
— Installing Components in the Base
— CPU and Expansion Unit Wiring Guidelines
— I/O Wiring Strategies
— I/O Module Wiring and Specifications
— Glossary of Specification Terms

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 or damage to 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 are 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 you 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 governments and state governments have additional requirements above and beyond those described in the NEC Handbook. Check with your local Electrical Inspector or Fire Marshall office for information.

Three Levels of Protection

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

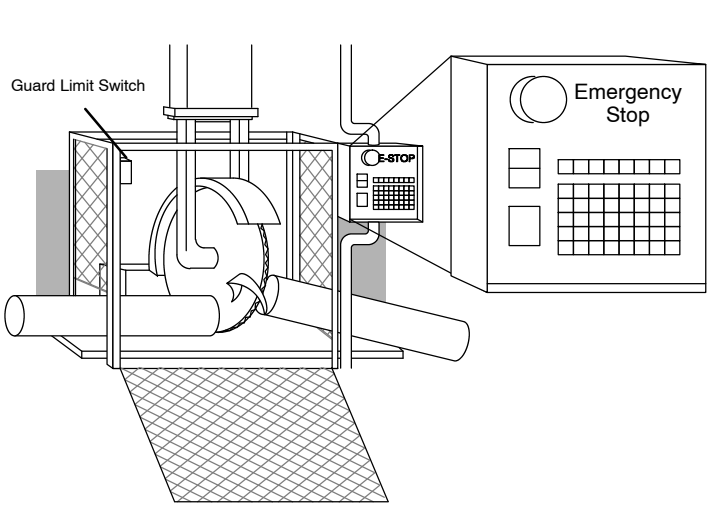
- Emergency stop switch for disconnecting system power
- Mechanical disconnect for output module power
- Orderly system shutdown sequence in the PLC control program

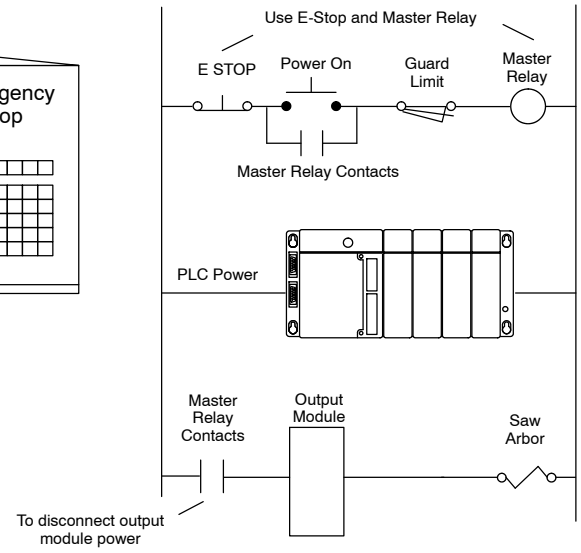
Emergency Stops

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

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

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





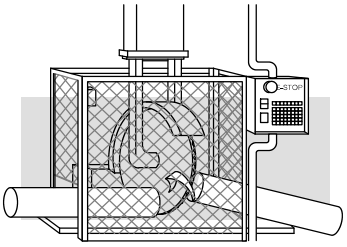
Emergency Power Disconnect

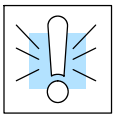
A properly rated emergency power disconnect should be used to power the PLC controlled system as a means of removing the power from the entire control system. It may be necessary to install a capacitor across the disconnect to protect against a condition known as “outrush”. This condition occurs when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

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

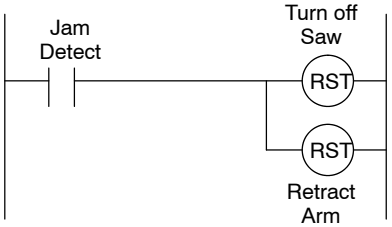
Orderly System Shutdown

Ideally, the first level of fault detection is the PLC control 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.



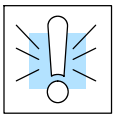


WARNING: The control program *must not* be the only form of protection for any problems that may result in a risk of personal injury or equipment damage.

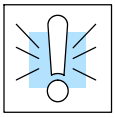


~~Class 1, Division 2 Approval~~

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



~~**WARNING: Explosion Hazard! – Substitution of components may impair suitability for Class 1, Division 2.**~~



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

See Errata Sheet at the beginning of this file. Delete this section. The DL405 system is not Class 1, Division 2 approved.

Mounting Guidelines

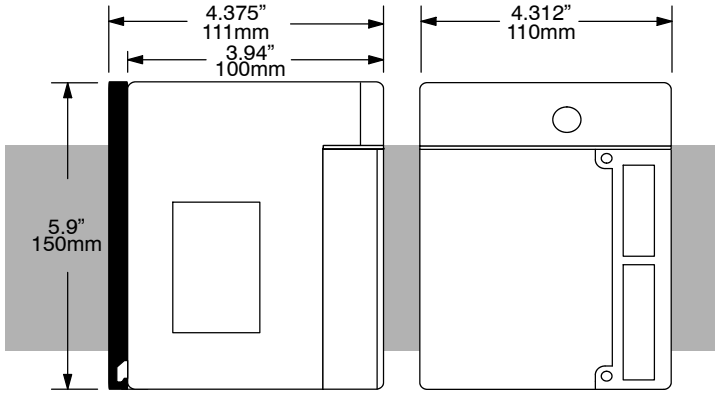
In addition to the panel layout guidelines, other specifications can affect the installation of a PLC system. Always consider the following:

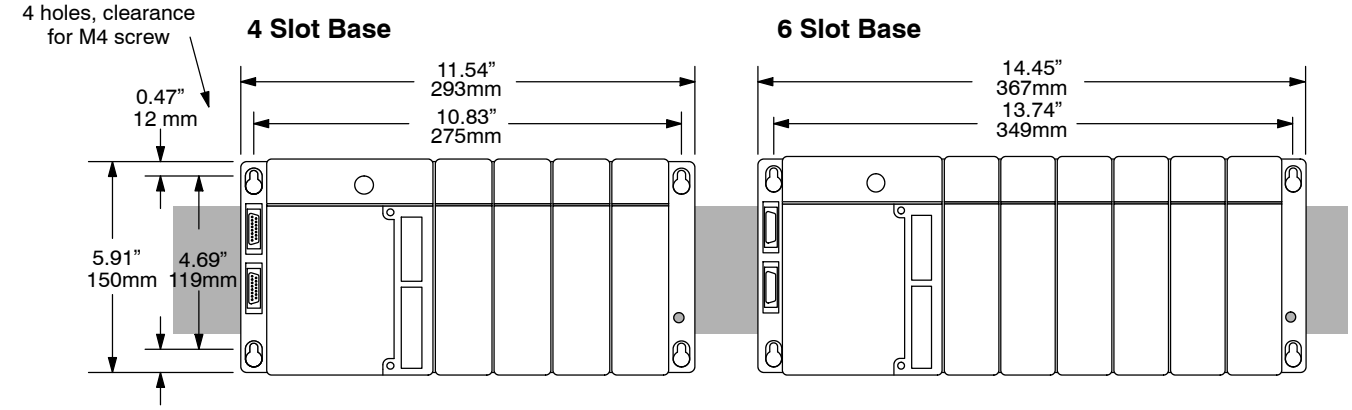
- Environmental specifications
- Power supply specifications
- Regulatory Agency Approvals
- Enclosure Selection and Component Dimensions

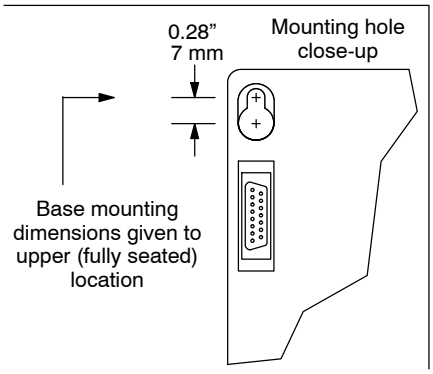
Base Dimensions

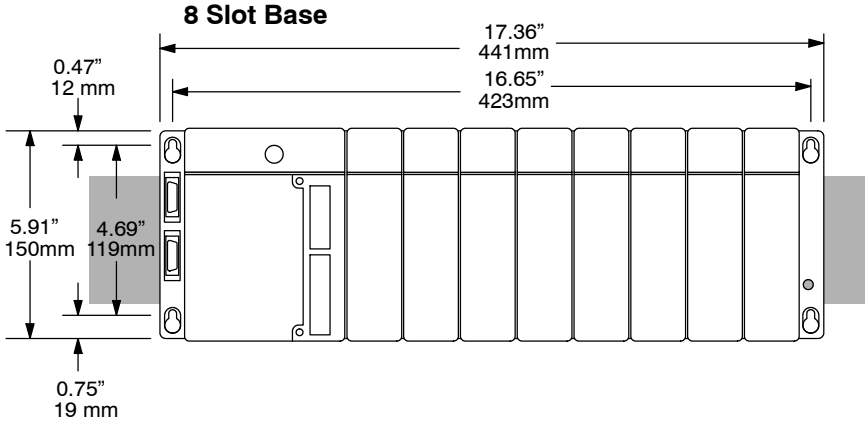
The following diagram shows the outside dimensions and mounting hole locations for the 4-slot, 6-slot, and 8-slot bases. Make sure you follow the installation guidelines to allow proper spacing from other components.

CPU Expansion Unit, Remote Slave





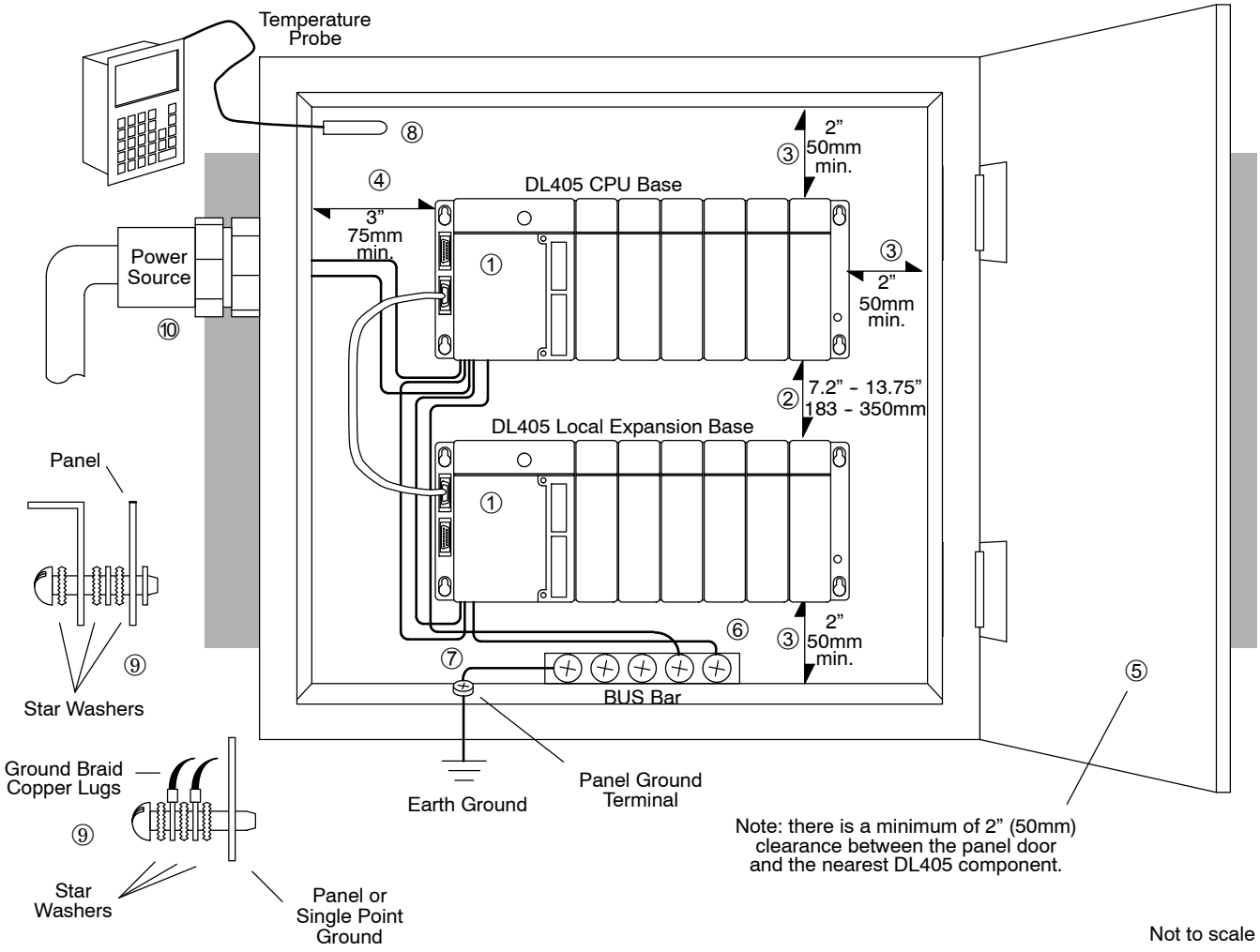




Panel Layout & Clearances

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

1. The bases must be mounted horizontally to provide proper ventilation.
2. There should be a minimum of 7.2" (183mm) and a maximum of 13.75" (350mm) between bases.
3. A minimum clearance of 2" (50mm) between the base and the top, bottom and right side of the cabinet should be provided.
4. A minimum clearance of 3" (75mm) between the base and the left side of the cabinet should be provided.
5. There must be a minimum of 2" clearance between the panel door and the nearest DL405 component.



6. Connect the ground terminal on the DL405 base to a single point ground. Use copper stranded wire to achieve a low impedance. Copper eye lugs should be crimped and soldered to the ends of the stranded wire to ensure good surface contact. Remove anodized finishes and use copper lugs and star washers at termination points. A rule of thumb is to achieve 0.1 ohm of DC resistance between the DL405 base and the single point ground.

7. 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 area.

   A good common ground reference (Earth ground) is essential for proper operation of the DL405, which include:

   a) Installing a ground rod as close to the panel as possible.
   b) Connection to incoming power system ground.

8. Installations where the ambient temperature may approach the lower or upper limits of the specifications should be evaluated properly. To do this 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.
9. 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 assure good contact on termination areas impediments such as paint, coating or corrosion should be removed in the area of contact.
10. 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, cooling, and maintenance

Agency Approvals

Some applications require agency approvals. The DL405 agency approvals for which DL405 products are submitted are;

- UL (Underwriters' Laboratories, Inc.)
- CE EMC (Electromagnetic Compatibility)
- CUL (Canadian Underwriters' Laboratories)

A complete listing of agency approvals for each product in the DL405 family is available in the sales catalog, or you may call 1–800–633–0405 (U.S.).

Environmental Specifications

The following table lists the environmental specifications that generally apply to the DL405 system (CPU, Expansion Unit, Bases, I/O Modules). The ranges that vary for the Handheld Programmer are noted at the bottom of this chart. I/O module operation may fluctuate depending on the ambient temperature and your application. Please refer to the appropriate I/O module chapters for the temperature derating curves applying to specific modules.

Specification	Rating
Storage temperature	-4° F to 158° F (-20° C to 70° C)*
Ambient operating temperature	32° F to 140° F (0° C to 60° C)
Ambient humidity	5% - 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

*Storage temperature for the Handheld Programmer is 14° to 149°F (-10° to 65° C)
**Ambient humidity for the Handheld Programmer is 20% to 90% non-condensing.

Power

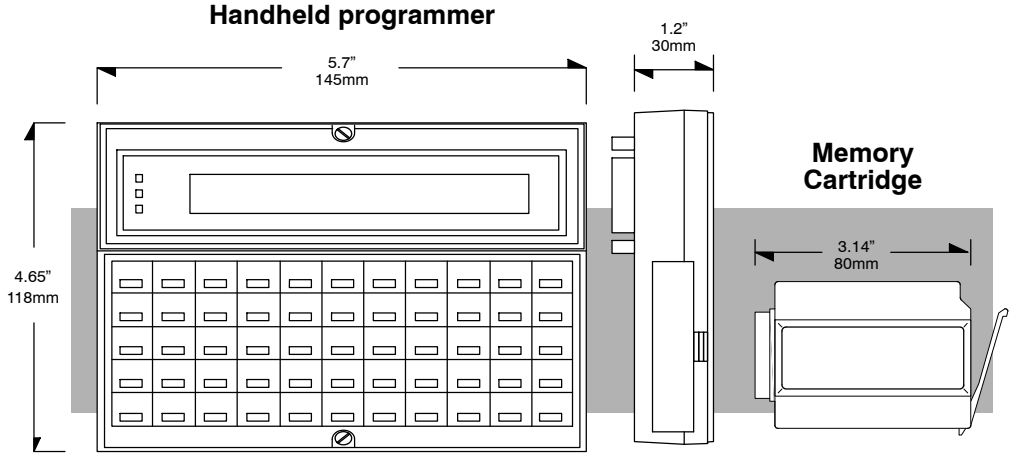
The external power source must be capable of suppling voltage and current complying with the PLC power supply specifications.

Specifications	DL405 Series CPUs
Voltage withstand (dielectric strength)	1 min. @ 1500 VAC between primary, secondary, field ground and run relay
Insulation resistance	> 10MΩ at 500 VDC
Input voltage range D4-430 / D4-440 / D4-450 / D4-EX	85-132 VAC (110 range) / 170-264 VAC (220 range)
Input voltage range D4-440DC-1 / D4-EXDC	20-29 VDC (24VDC) less than 10% ripple
Input voltage range D4-440DC-2 / D4-EXDC-2	90-146 VDC (125 VDC) less than 10% ripple
Maximum inrush current D4-430 / D4-440 / D4-EX	20A
Maximum inrush current D440DC-1 / D4-EXDC	10A
Maximum inrush current DL440DC-2 / D4-EXDC-2	20A
Maximum power DL430/DL440/DL450, D4-EX	50VA
Maximum power DL440DC-1, D4-EXDC	38W
Maximum power DL440DC-2, D4-EXDC-2	30W
24VDC Auxiliary Power Supply (D4-EX only)	20-28 VDC @ 0.4A maximum, ripple > 1V p-p

Component Dimensions

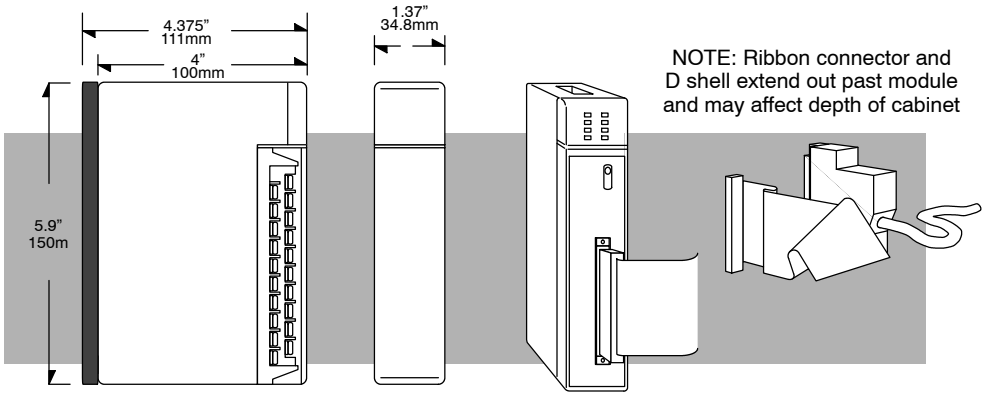
Before installing your PLC system you will need to know the dimensions for the components in your system. The diagram on this page provide the component dimensions and should be used to define your enclosure specifications. Remember to leave room for potential expansion. Appendix E provides the weights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

Handheld programmer



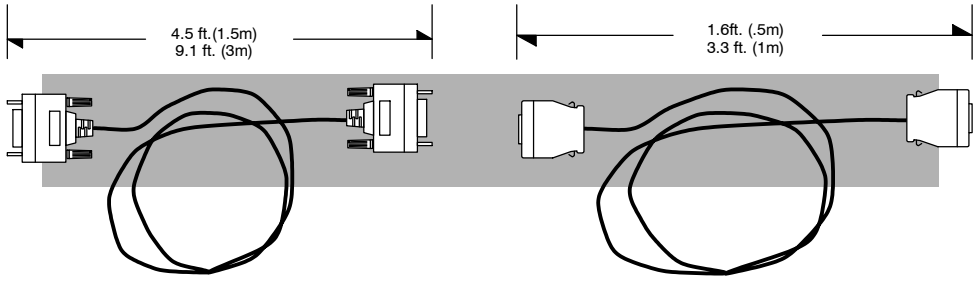
I/O modules

I/O module w/Ribbon connector



Handheld programmer cable

Base Expansion Cable



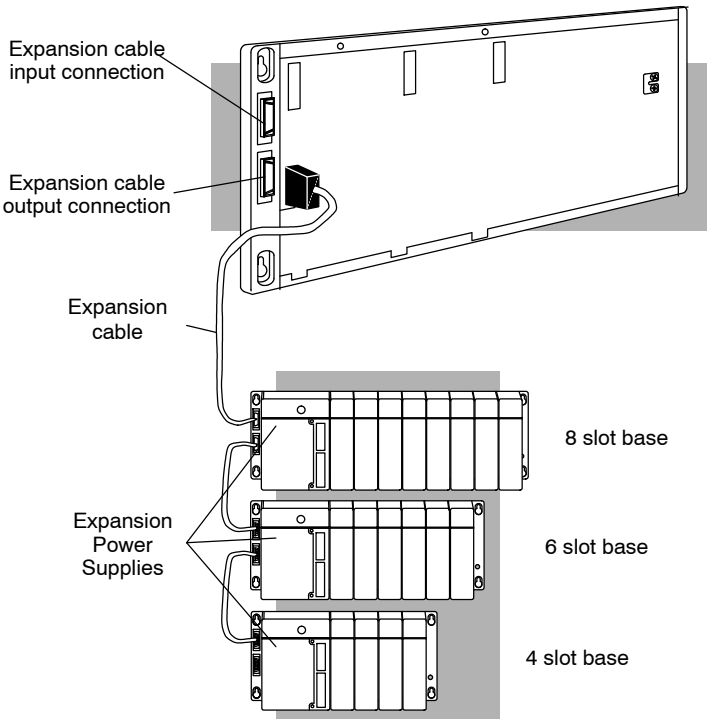
Installing DL405 Bases

Three Sizes of Bases

All I/O configurations of the DL405 (except for Slice I/O) 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.

Local Base

Expansion Bases

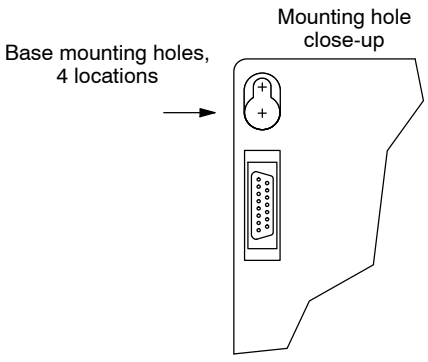


WARNING: To minimize the risk of electrical shock, personal injury, or equipment damage, always disconnect the system power before installing or removing any system component.

Mounting the Base

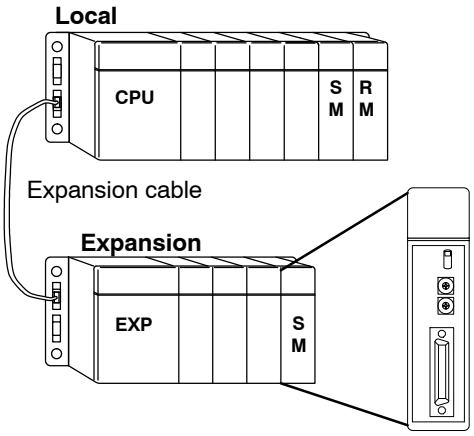
The CPU/Expansion Unit/Remote Slave must always be installed in the left-most slot in a base. This slot is marked on the base as P/S, CPU. The I/O modules can be installed in any remaining slots. It is not necessary for all slots to be filled for your system to work correctly. You may use filler modules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



Choosing the Base Type

There are two types of bases to choose from. The standard base type restricts the placement of specialty modules (or intelligent modules) to the local base with the CPU. By using the DL450 CPU and the new "expanded bus" base type, you can also use specialty modules in expansion bases as shown to the right. When all bases in the local/expansion system are of the new type, the DL450 can communicate with specialty modules in any base. In all other respects, the new base is an exact replacement for the standard bases.



The part numbers for standard bases and the new bases are listed below.

Standard Bases

D4-04B

D4-06B

D4-08B

Expanded Bus Bases

D4-04B-1

D4-06B-1

D4-08B-1

Allows selected specialty modules in expansion bases

The base expansion connectors on the new bases have new data signals used in communicating with specialty I/O across bases. Accordingly, you must observe the following restrictions and guidelines with the new bases:

- Only the DL450 type CPU (in the local base) can communicate with a specialty module in an expansion base.
- In the above case, both local and expansion bases must be the new (-1) type.
- Of course, you can still have specialty modules in the local base.
- The new bases can also be used with DL430 and DL440 CPUs (however, these CPUs cannot communicate with specialty I/O in expansion bases).
- You can mix standard bases with new bases in a system, but no specialty I/O modules may be used in expansion bases in this case (the standard bases do not pass through the specialty I/O signals on their expansion connectors).

NOTE: If you are designing a new DL450 CPU-based application, we recommend using the new bases (-1 type) so you can add specialty modules in any base later.

Installing Components in the Base

Setting the CPU DIP Switches (DL430/440 Only)

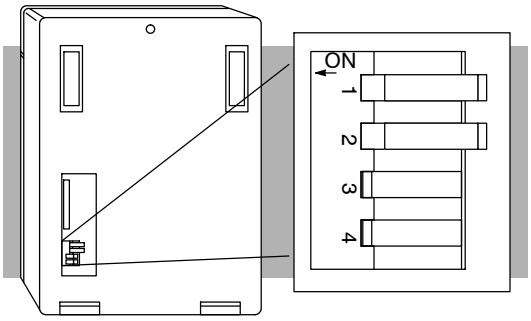
There is one bank of four configuration switches located on the back of DL430 and DL440 CPUs. These switches affect battery low detection, station address override and baud rate of the secondary port (25-pin D connector). The figure below indicates the location of these DIP switches. Equivalent configuration of the DL450 CPU requires selecting AUX functions on a programming device.

Switch 1

- ON = Battery low indicator disabled
- OFF= Battery low indicator enabled

Switch 2

- ON = Station address override is enabled (address 1)
- OFF= Station address is set by AUX function with programming device

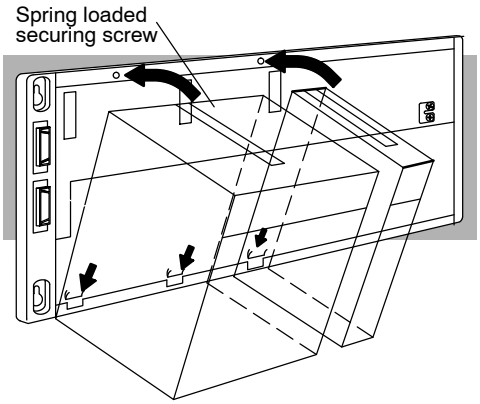


NOTE: Setting Switch 2 on forces the station address to 1. It does not change the address set by the programming device. When Switch 2 is turned off again the address will revert back to the address stored in memory via the AUX function.

Port 1 Baud Rate	Switch 3	Switch 4
300	Off	Off
1200	Off	On
9600	On	Off
19200	On	On

NOTE: Parity, Mode and Station address for port 2 is selected by AUX functions using a programming device.

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

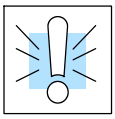


WARNING: To minimize the risk of electrical shock, personal injury, or equipment damage, always disconnect the system power before installing or removing any system component.

CPU and Expansion Unit Wiring Guidelines

The main power terminal connections are under the front covers of the DL405 CPUs and 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.

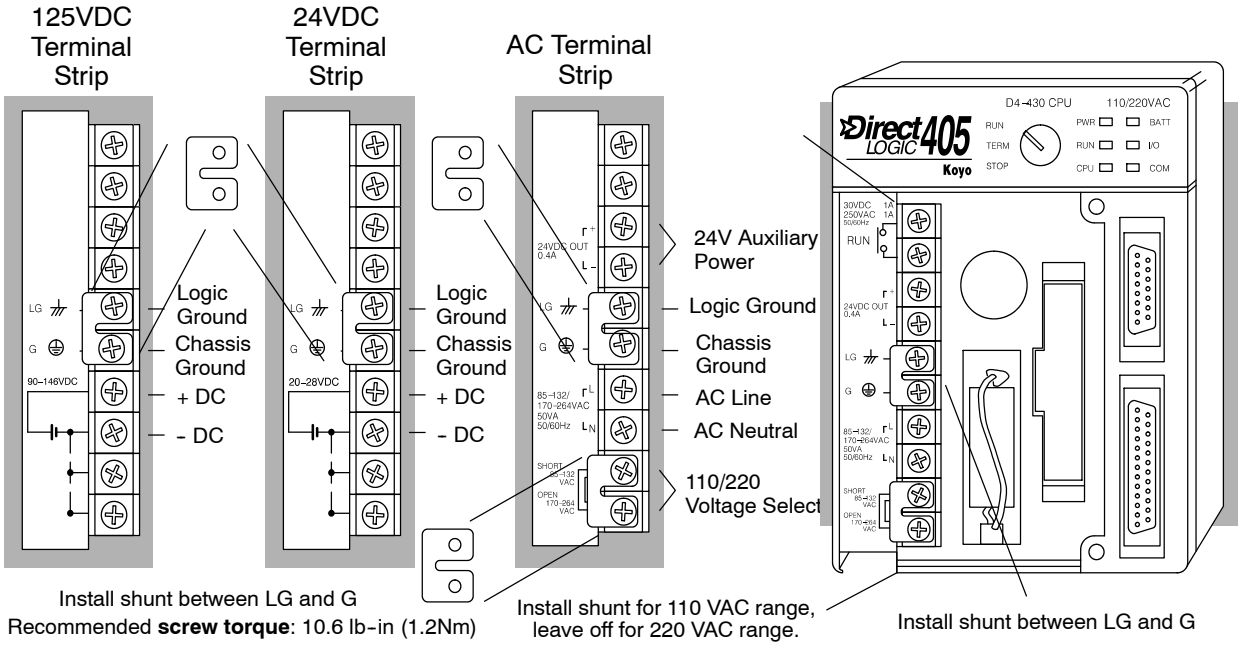
- **Run Relay** – (CPU only) indicates to an external device when the CPU is in Run Mode by contact closure. Its normally-open contacts can also remove power from critical I/O points if CPU comes out of Run mode.
- **24VDC Auxiliary Power** – can be used to power field devices or I/O modules requiring external power. It supplies up to 400 mA of current at 20–28VDC, ripple less than 1 V P-P. (Not available on DC CPUs.)
- **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 24/125 VDC CPU. The positive connection is tied to line and the negative connection is tied to ground.)
- **110/220 Voltage Select** – a shunt across two of the terminals determines the voltage selection. Install the shunt to select 110VAC input power, and remove the shunt to select 220VAC power input (the shunt is not required for DC-powered CPUs or Expansion Units.)



WARNING: Damage will occur to the power supply if 220 VAC is connected to the terminal connections with the 115 VAC shunt installed. Once the power wiring is connected, install the protective cover to avoid risk of accidental shock.

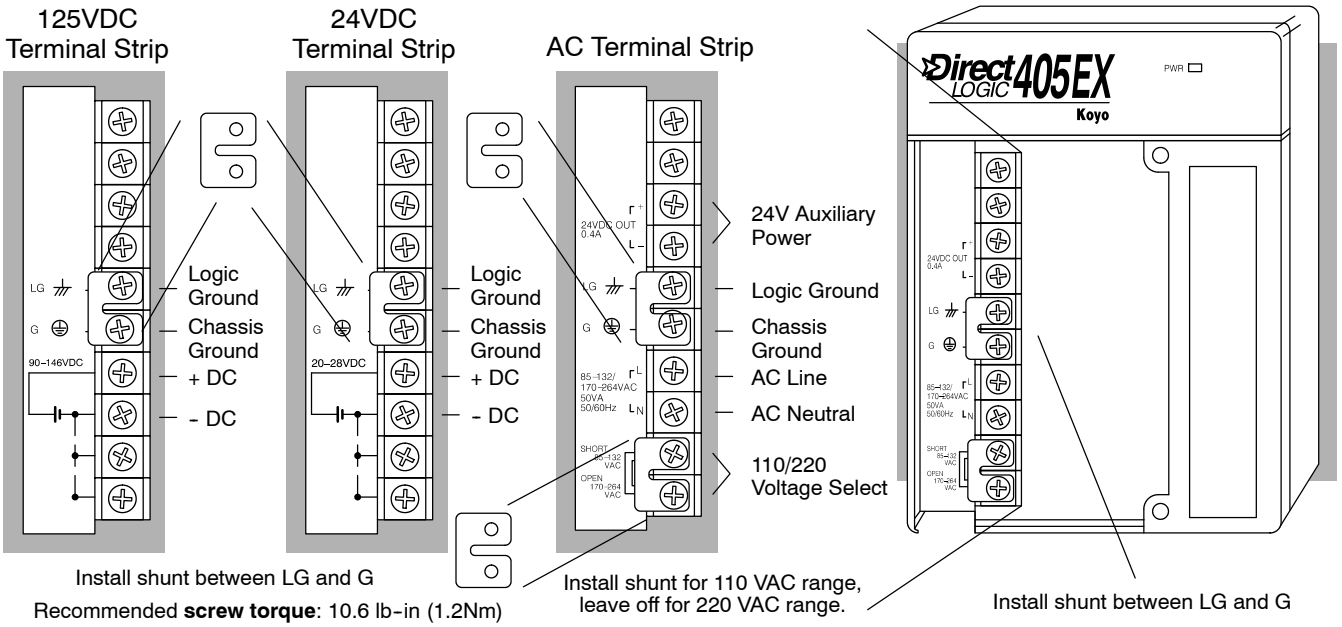
CPU Wiring

The following diagram details the appropriate connections for each terminal.



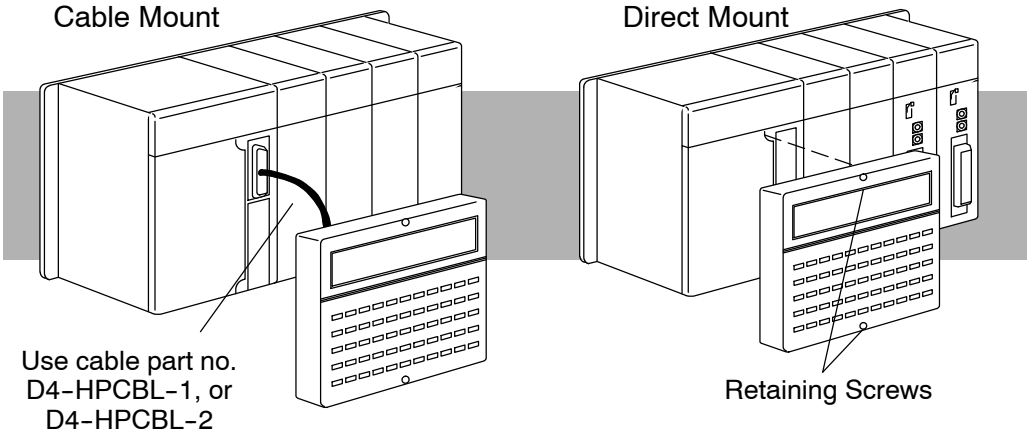
Expansion Unit Wiring

The following diagram details the appropriate connections for each terminal.

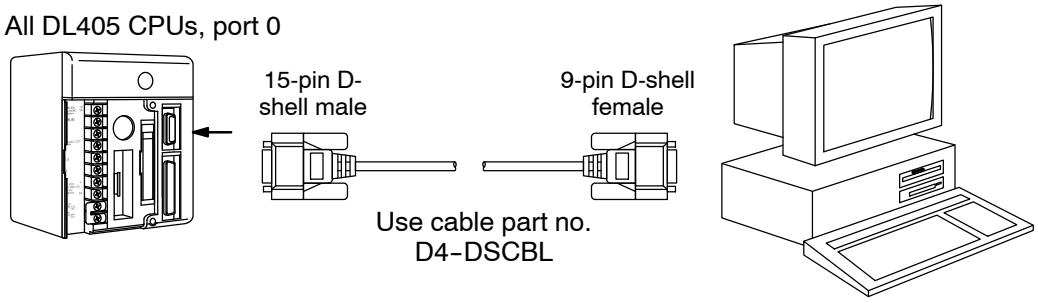


Connecting Programming Devices

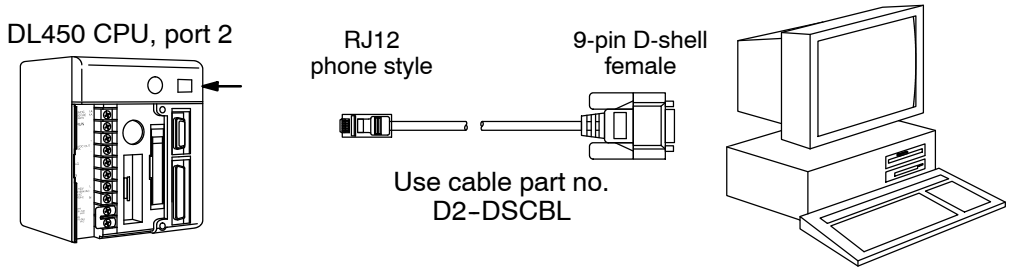
You can mount the Handheld directly to Port 0 of any DL405 CPU (15-pin D-shell connector), or you can use a 9 foot (3m) or 4.6 ft (1.5m) cable as shown below.



The standard port for use in ***Direct***SOFT programming is the 15-pin port 0 on all DL405 CPUs. The cable shown below is approximately 12 feet (3.66m) long.

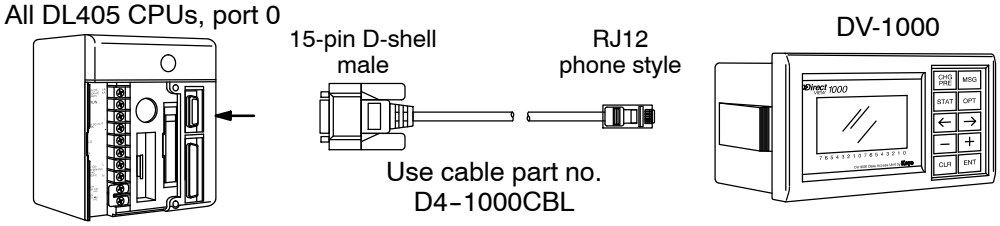


On the DL450, you may use port 2 instead for ***Direct***SOFT programming. The cable shown below is approximately 12 feet (3.66m) long.

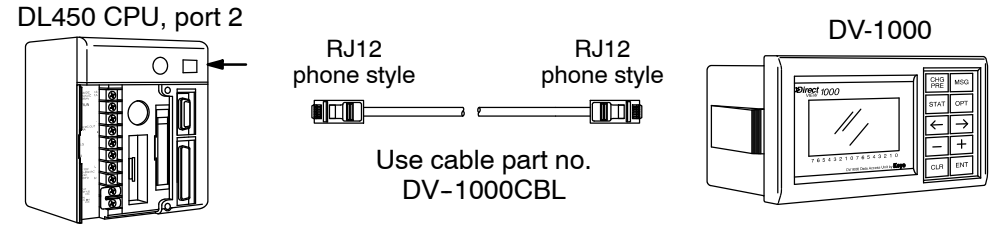


Connecting Operator Interface Devices

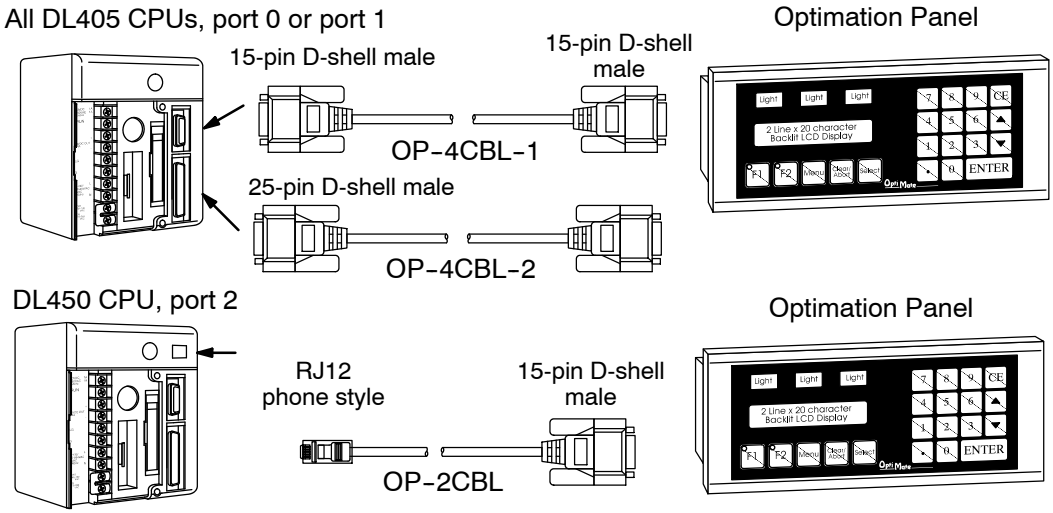
Operator interfaces usually require data and power connections. However, the popular DV-1000 Data Access Unit may receive data and power directly from any DL405 CPU, using the 2 meter (6.56 ft.) long cable shown below.



The DL450 can connect to a DV-1000 from port 2, using the 2 meter (6.56 ft.) long cable shown below.



Optimation operator interface panels require separate power and data connections. Connect the CPU port 0, port 1, or port 2 (DL450) to an Optimation panel choosing the appropriate 2 meter (6.56 ft.) long cable from the three shown below.

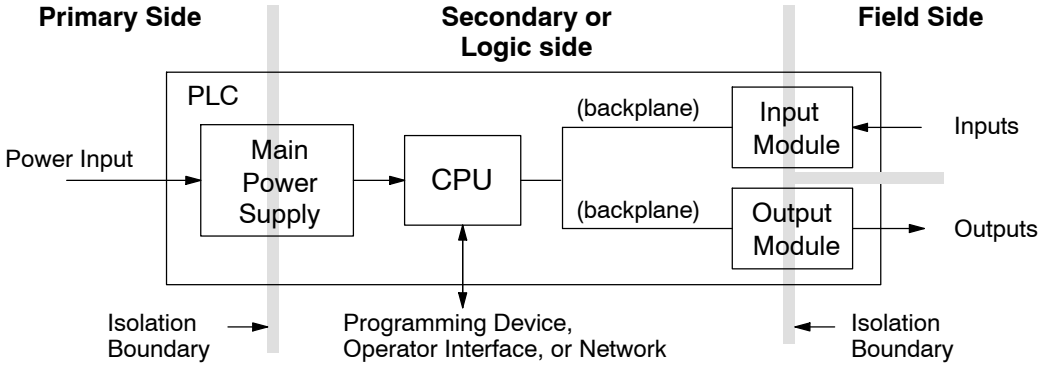


I/O Wiring Strategies

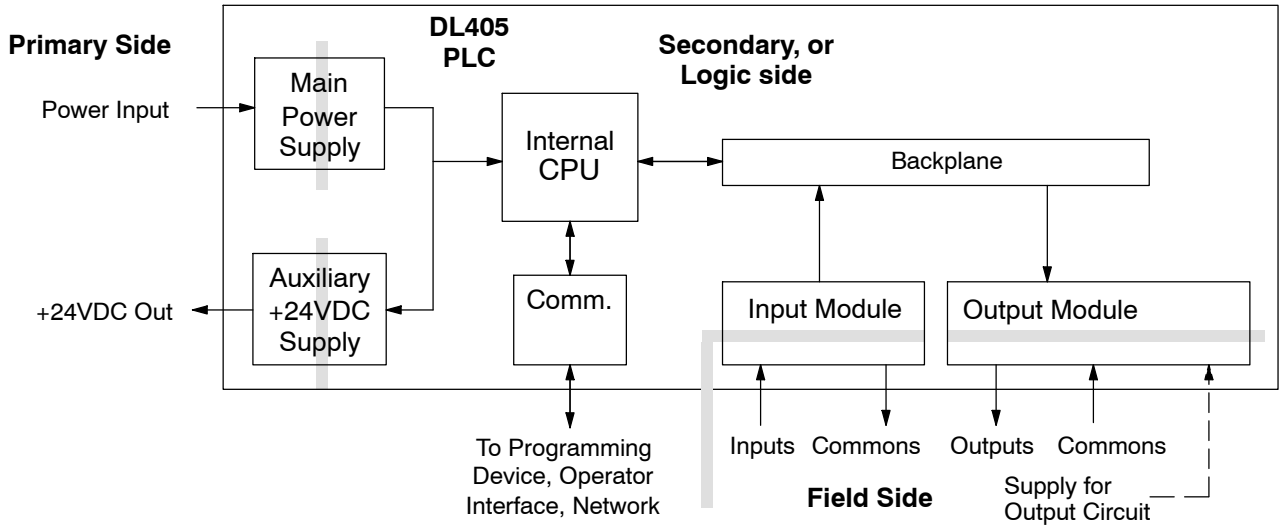
The DL405 PLC system is very flexible and will work in many different wiring configurations. By studying this section before actual installation, you can probably find the best wiring strategy for your application . This will help to lower system cost, wiring errors, and avoid safety problems.

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 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 that 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.*



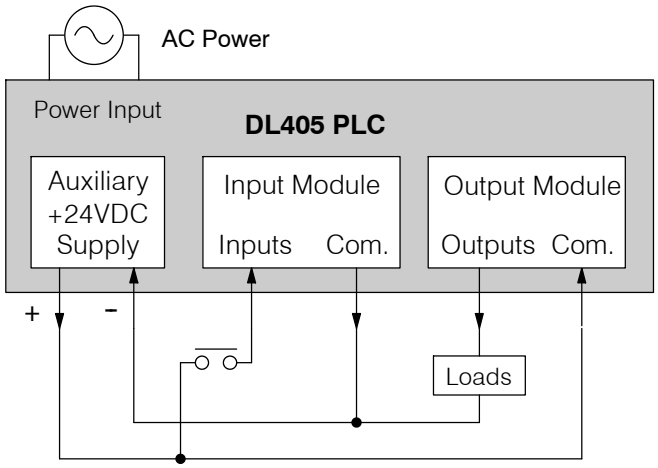
The next figure shows the physical layout of a DL405 PLC system, as viewed from the front. In addition to the basic circuits covered above, AC-powered CPUs include an auxiliary +24VDC power supply with its own isolation boundary. Since the supply output is isolated from the other three circuits, it can power input and/or output circuits!



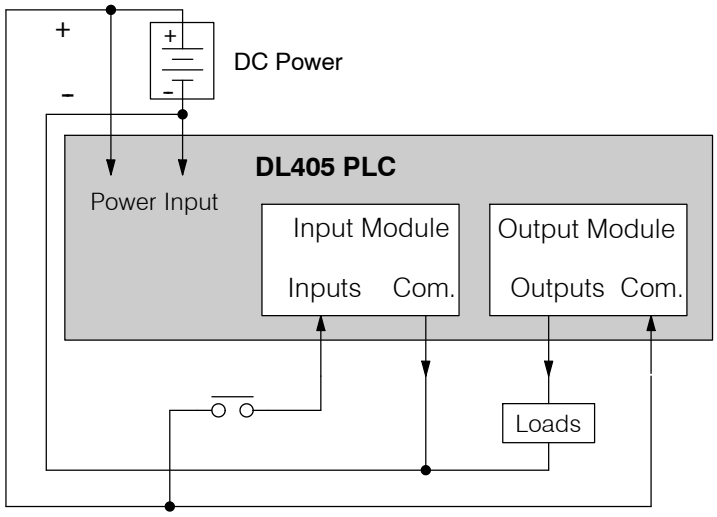
In some cases, using the built-in auxiliary +24VDC supply can result in a cost savings for your control system. It can power combined loads up to 400 mA. Be careful not to exceed the current rating of the supply. If you are the system designer for your application, you may be able to select and design in field devices which can use the +24VDC auxiliary supply.

Powering I/O Circuits with the Auxiliary Supply

All DL405 CPUs feature the internal auxiliary supply. If input devices AND output loads need +24VDC power, the auxiliary supply may be able to power both circuits as shown in the following diagram (400 mA limit).



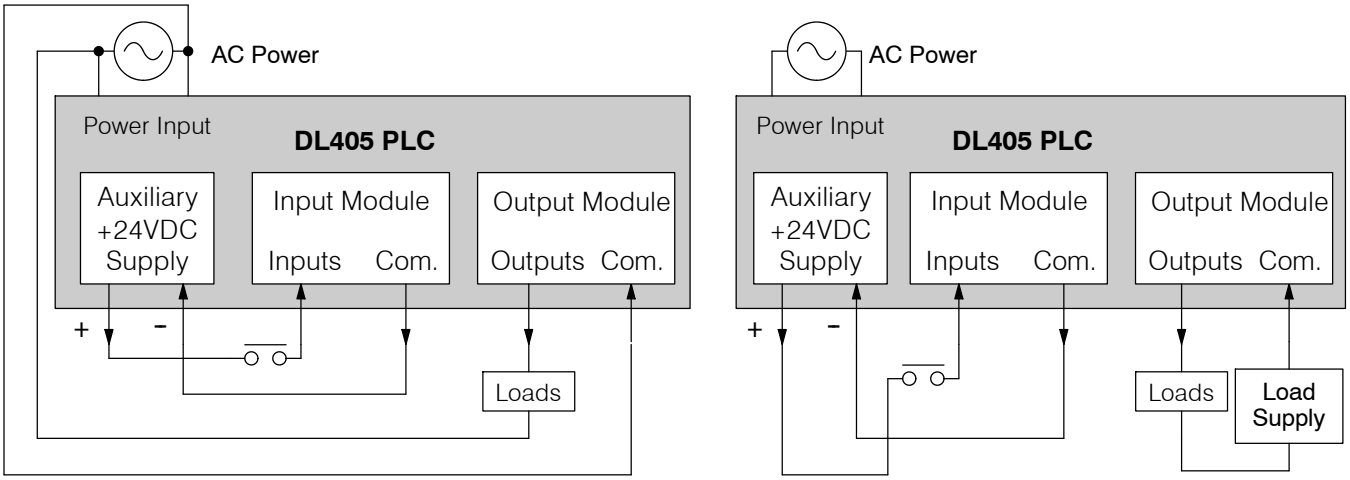
DC-powered DL405 CPUs are designed for application environments in which low-voltage DC power is more readily available than AC. These include a wide range of battery–powered applications, such as remotely-located control, in vehicles, portable machines, etc. For this application type, all input devices and output loads typically use the same DC power source. Typical wiring for DC-powered applications is shown in the following diagram.



Powering I/O Circuits Using Separate Supplies

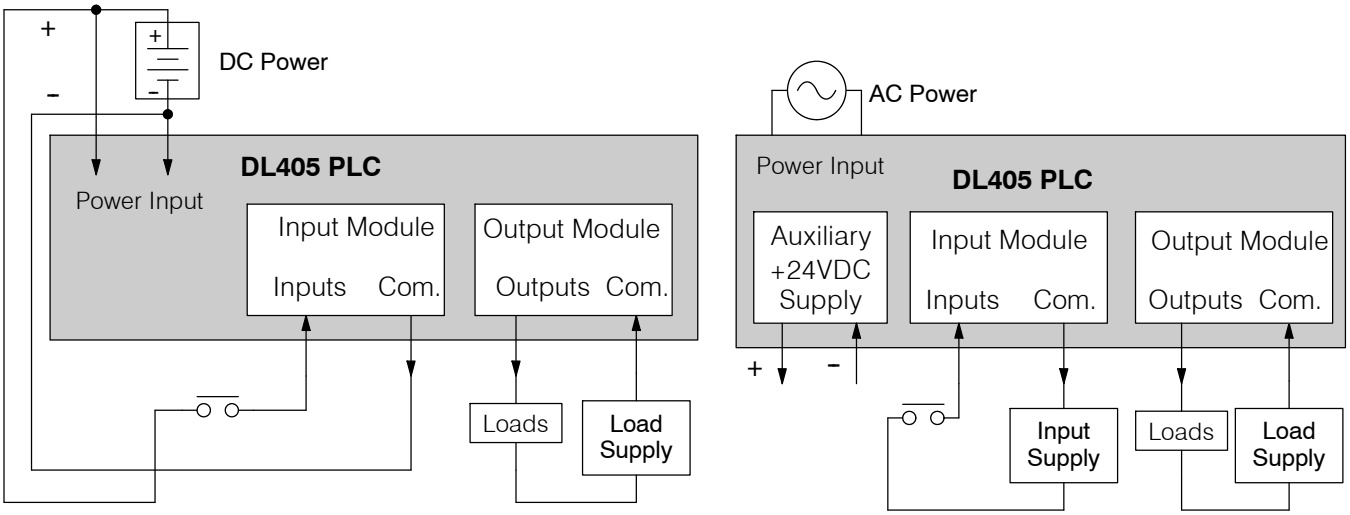
In most applications it will be necessary to power the input devices from one power source, and to power output loads from another source. Loads often require high-energy AC power, while input sensors use low-energy DC. If a machine operator is likely to come in close contact with input wiring, then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

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



Some applications will use the PLC external power source to also power the input circuit. This typically occurs on DC-powered PLCs, as shown in the drawing below to the left. The inputs share the PLC power source supply, while the outputs have their own separate supply.

A worst-case scenario, from a cost and complexity view-point, 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 that the auxiliary supply output is an unused resource. For these reasons, you'll probably want to avoid this situation if possible.



Sinking/Sourcing Concepts

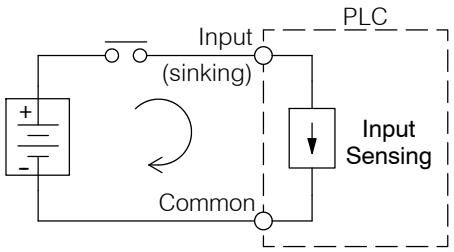
Before going further in our study of wiring strategies, we must have a solid understanding of "*sinking*" and "*sourcing*" concepts. Use of these terms occurs frequently in input or output circuit discussions. It is the goal of this section to make these concepts easy to understand, further ensuring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

Sinking = provides a path to supply **ground (-)**

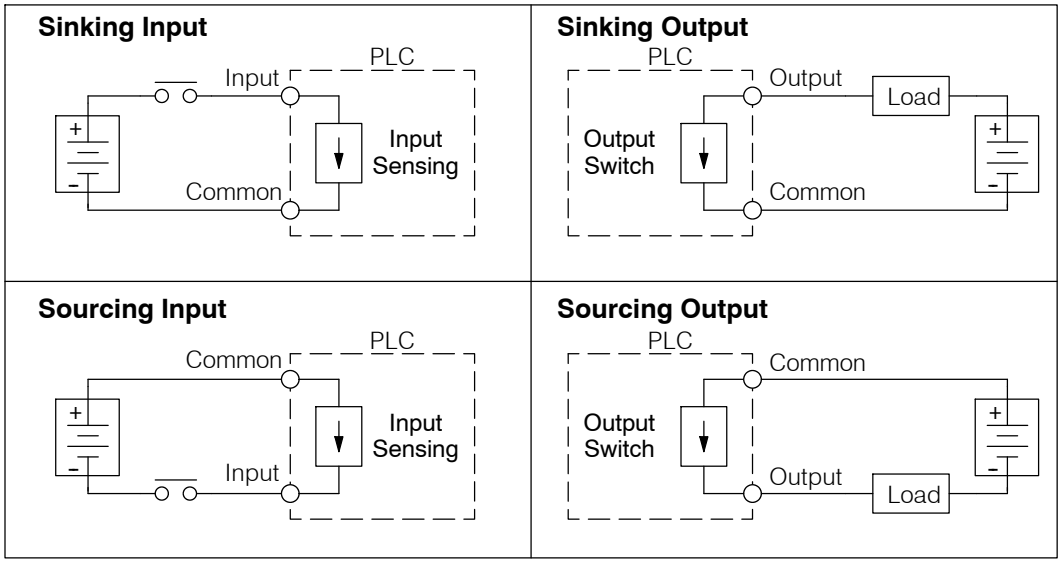
Sourcing = provides a path to supply **source (+)**

First you will notice that 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 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, we 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, we just have to connect it so the input *provides a path to ground (-)*. So, we 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, we have completed the circuit. Current flows in the direction of the arrow when the switch is closed.

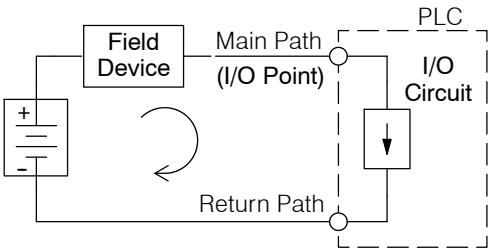


By applying the circuit principle above to the four possible combinations of input/output sinking/sourcing types, we have the four circuits as shown below. The I/O module specifications at the end of this chapter list the input or output type.

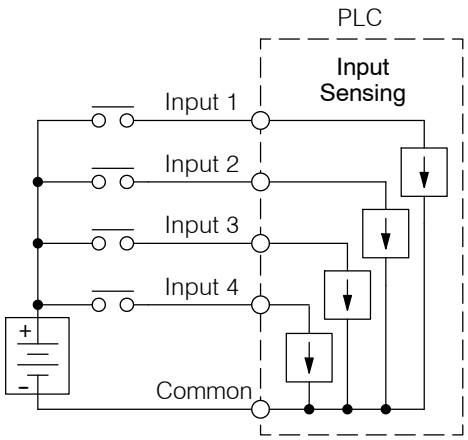


I/O "Common" Terminal Concepts

In order for a PLC I/O circuit to operate, current must enter at one terminal and exit at another. This means 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.



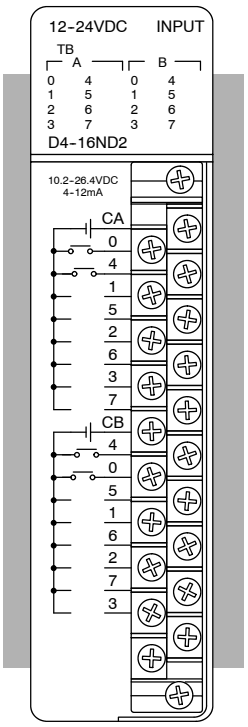
If we had unlimited space and budget for I/O terminals, then every I/O point could have two dedicated terminals just as the figure above shows. However, providing this level of flexibility is not practical or even necessary for most applications. So, most Input or Output points on PLCs are in groups which share the return path (called *commons*). The figure to the right shows a group (or *bank*) of 4 input points which share a common return path. In this way, the four inputs require only five terminals instead of eight.

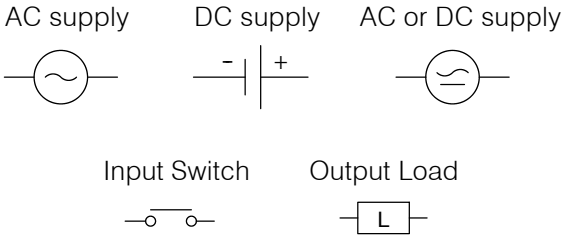


NOTE: In the circuit above, the current in the common path is 4 times any channel's input current when all inputs are energized. This is especially important in output circuits, where heavier gauge wire is sometimes necessary on commons.

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. The miniature schematic shows 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:



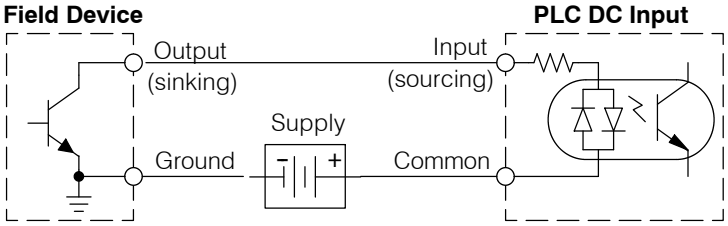


Connecting DC I/O to "Solid State" Field Devices

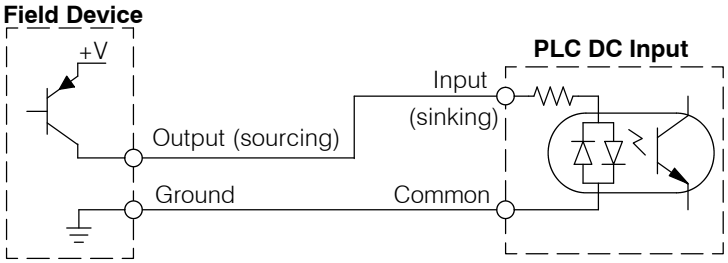
In the previous section on Sourcing and Sinking concepts, we explained that DC I/O circuits sometimes will only allow current to flow one way. This is also true for many of the field devices which have solid-state (transistor) interfaces. In other words, field devices can also be sourcing or sinking. *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 in that they detect current flow in either direction, so they can be wired as either sourcing or sinking. In the following circuit, a field device has an open-collector NPN transistor output. It sinks current from the PLC input point, which sources current. The power supply can be the +24 auxiliary supply or another supply (+12 VDC or +24VDC), as long as the input specifications are met.



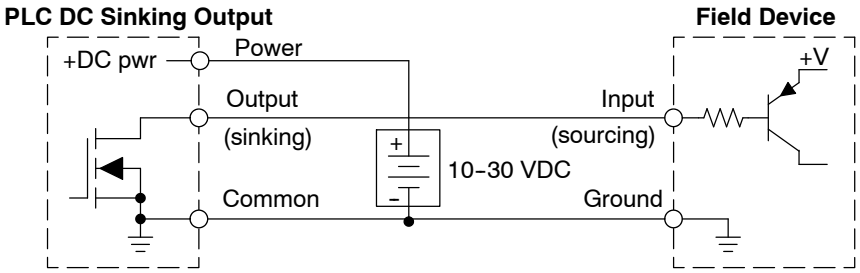
In the next circuit, a field device has an open-emitter PNP transistor output. It sources current to the PLC input point, which sinks the current back to ground. Since the field device is sourcing current, no additional power supply is required.



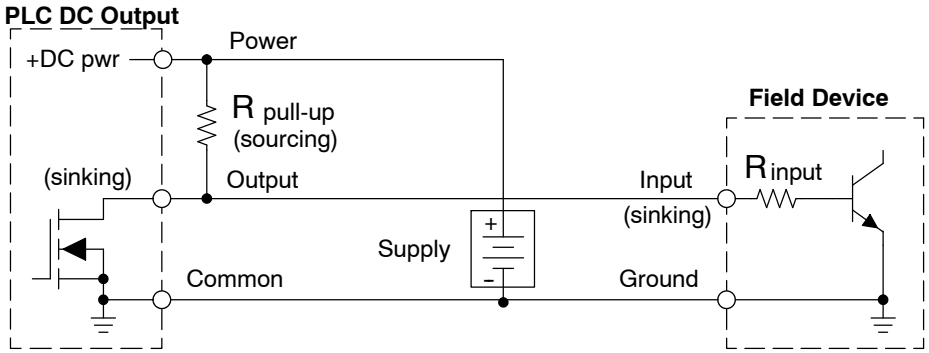
Solid State Output Loads

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



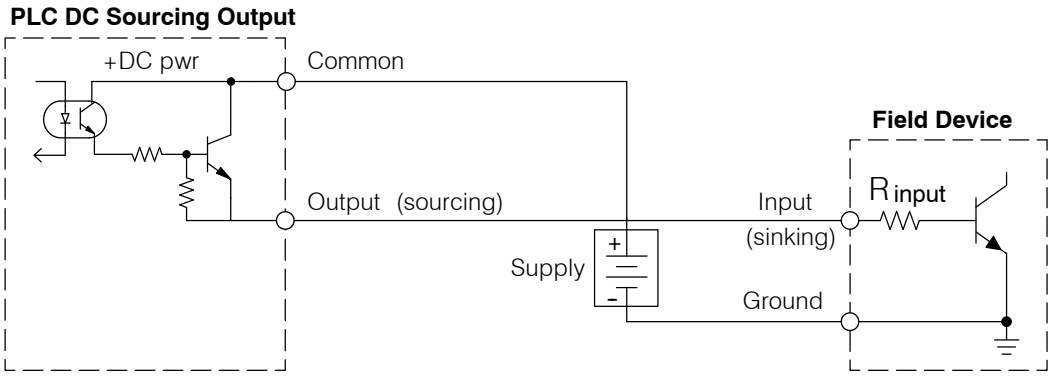
NOTE 1: DO NOT attempt to drive a heavy load (>25 mA) with this pull-up method
NOTE 2: Using the pull-up resistor to implement a sourcing output has the effect of inverting the output point logic. In other words, the field device input is energized when the PLC output is OFF, from a ladder logic point-of-view.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

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

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

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

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



Relay Output Guidelines

Four 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)

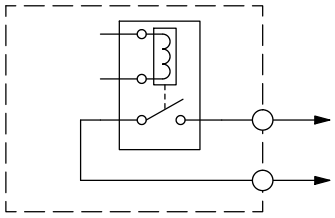
Some applications in which NOT to use relays:

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

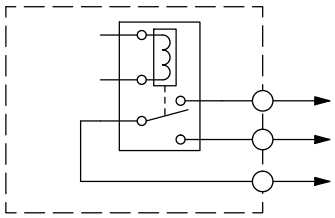
Relay outputs in the DL405 output modules are available in two contact arrangements, shown to the right. The Form A type, or SPST (single pole, single throw) type is normally open and is the simplest to use. The Form C type, or SPDT (single pole, double throw) type has a center contact which moves and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



Transient Suppression for Inductive Loads in a Control System

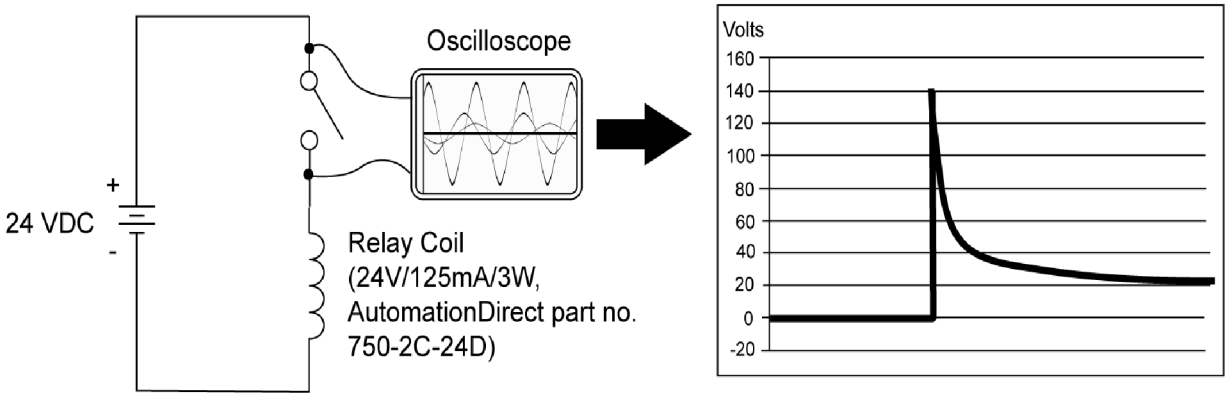
The following pages are intended to give a quick overview of the negative effects of transient voltages on a control system and provide some simple advice on how to effectively minimize them. The need for transient suppression is often not apparent to the newcomers in the automation world. Many mysterious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

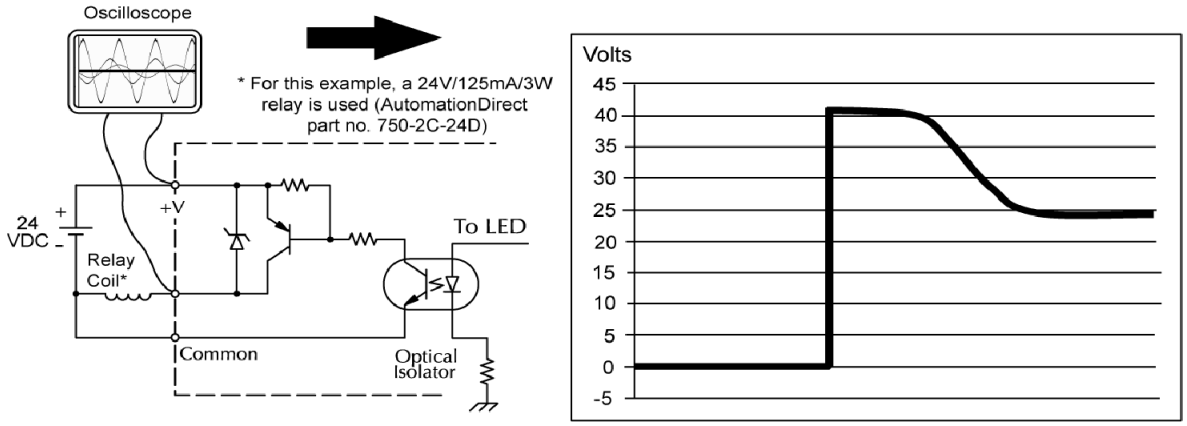
A very destructive side-effect of the arcing across relay contacts is the electromagnetic interference (EMI) it can cause. This occurs because the arcing causes a current surge, which releases RF energy. The entire length of wire between the relay contacts, the coil, and the power source carries the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

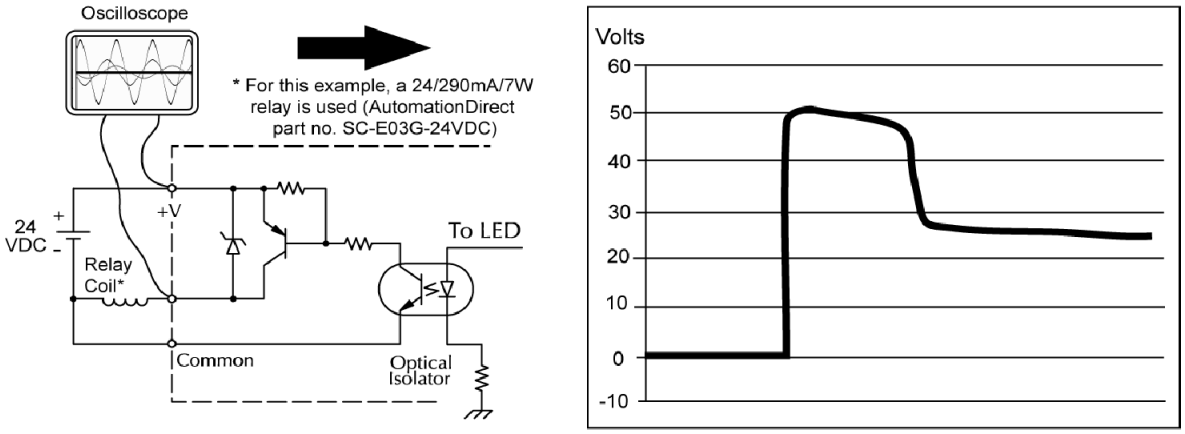
Here is another example using the same 24V/125mA/3W relay used earlier. This example measures the PNP transistor output of a D0-06DD2 PLC, which incorporates an integrated Zener diode for transient suppression. Instead of the 140V peak in the first example, the transient voltage here is limited to about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

Example: Small Inductive Load with Only Integrated Suppression



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

Example: Larger Inductive Load with Only Integrated Suppression

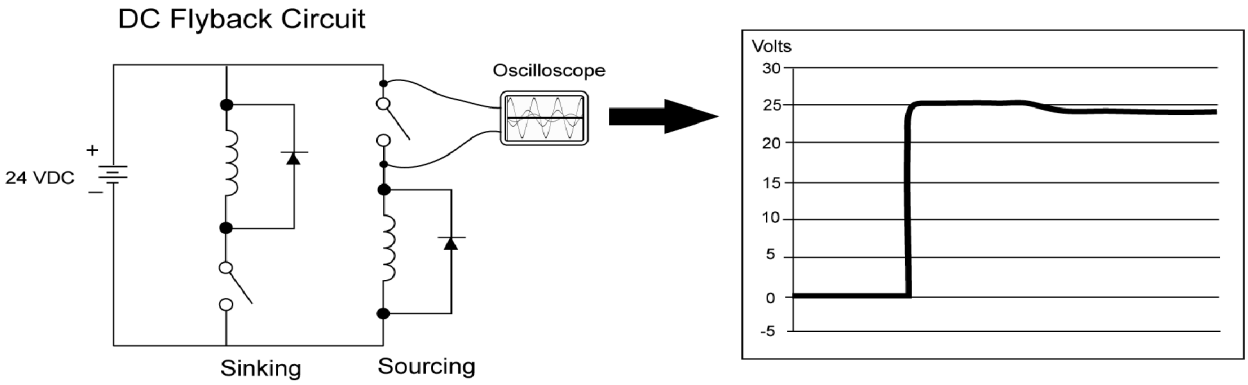


Additional transient suppression should be used in both of the preceding examples. If you are unable to measure the transients generated by the connected loads of your control system, using additional transient suppression on all inductive loads would be the safest practice.

Types of Additional Transient Protection

DC Coils:

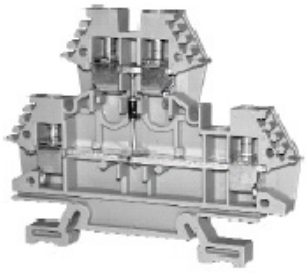
The most effective protection against transients from a DC coil is a flyback diode. A flyback diode can reduce the transient to roughly 1V over the supply voltage, as shown in this example.



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

AD-ASMD-250
Protection Diode Module

784-4C-SKT-1
Relay Socket



DN-D10DR-A
Diode Terminal Block

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

AutomationDirect's ZL-TSD8-24 transorb module is a good choice for 24 VDC circuits. It is a bank of 8 uni-directional 30 V 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

AC Coils:

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

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

ZL-TSD8-120
Transorb Module

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

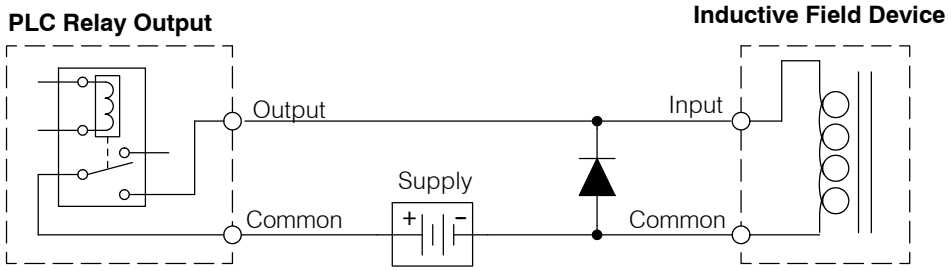
For example, a MOV or TVS diode rated for use on 24-48 VDC coils would need to have a high enough voltage rating to NOT conduct at 48V. That suppressor might typically start conducting at roughly 60VDC. If it were mounted across a 24V coil, transients of roughly 84V (if sinking output) or -60V (if sourcing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

Prolonging Relay Contact Life

Relay contacts wear according to the amount of relay switching, amount of spark created at the time of open or closure, and presence of airborne contaminants. There are some steps you can take to help prolong the life of relay contacts, such as switching the relay on or off only when it is necessary, and if possible, switching the load on or off at a time when it will draw the least current. Also, take measures to suppress inductive voltage spikes from inductive DC loads such as contactors and solenoids.

For inductive loads in DC circuits we recommend using a suppression diode as shown in the following diagram (DO NOT use this circuit with an AC power supply). When the load is energized the diode is reverse-biased (high impedance). When the load is turned off, energy stored in its coil is released in the form of a negative-going voltage spike. At this moment the diode is forward-biased (low impedance) and shunts the energy to ground. This protects the relay contacts from the high voltage arc that would occur just as the contacts are opening.

Place the diode as close to the inductive field device as possible. Use a diode with a peak inverse voltage rating (PIV) at least 100 PIV, 3A forward current or larger. Use a fast-recovery type (such as Schottky type). DO NOT use a small-signal diode such as 1N914, 1N941, etc. Be sure the diode is in the circuit correctly before operation. If installed backwards, it short-circuits the supply when the relay energizes.



Another method of surge suppression is to use a resistor and capacitor (RC) snubber network. The RC network must be located close to the relay module output connector. To find the values for the RC snubber network, first determine the voltage across the contacts when open, and the current through them when closed. If the load supply is AC, then convert the current and voltage values to peak values:

Now we are ready to calculate values for R and C, according to the formulas:

$$C\ (\mu F) = \frac{I^{2}}{10} \qquad R\ (\Omega) = \frac{V}{10 \times I^{x}}\ \text{, where } x = 1 + \frac{50}{V}$$

C minimum = 0.001 μF, the voltage rating of C must be ≥ V, non-polarized

R minimum = 0.5 Ω, 1/2 W, tolerance is ± 5%

For example, suppose a relay contact drives a load at 120VAC, 1/2 A. Since this example has an AC power source, we first, we calculate the peak values:

$$I_{peak} = I_{rms} \times 1.414, = 0.5 \times 1.414 = 0.707 \text{ Amperes}$$

$$V_{peak} = V_{rms} \times 1.414 = 120 \times 1.414 = 169.7 \text{ Volts}$$

Now, finding the values of R and C, we have:

$$C\ (\mu F) = \frac{I^{2}}{10} = \frac{0.707^{2}}{10} = 0.05\ \mu F, \text{ voltage rating} \geq 170 \text{ Volts}$$

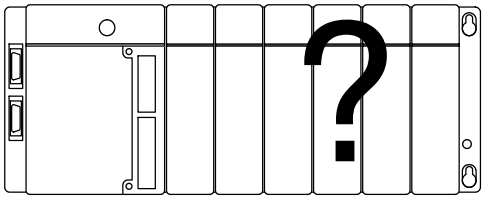
$$R\ (\Omega) = \frac{V}{10 \times I^{x}}\ \text{, where } x = 1 + \frac{50}{V}$$

$$x = 1 + \frac{50}{169.7} = 1.29 \qquad R\ (\Omega) = \frac{169.7}{10 \times 0.707^{1.29}} = 16\ \Omega,\ 1/2\ W,\ \pm\ 5\%$$

I/O Module Wiring and Specifications

Module Placement

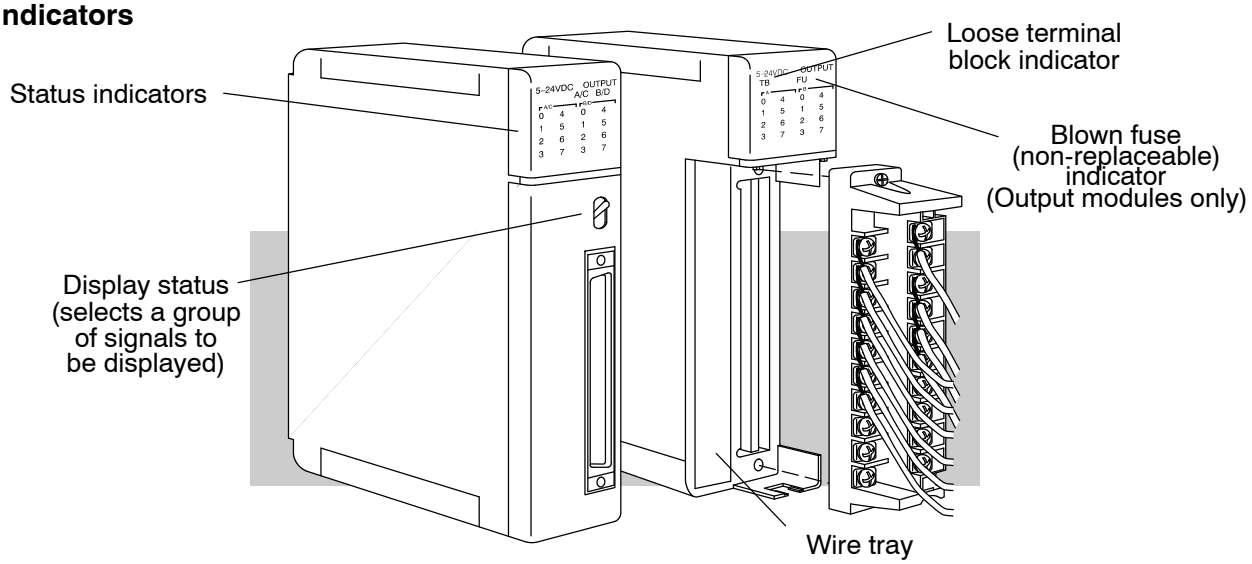
Before wiring the I/O modules in your system to field devices, it's very important to make sure each I/O module is in the right slot and base in the system. Costly wiring errors may be avoided by doing the following:



- 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 manuals before installation and wiring.
- Whenever possible, keep modules with high voltage and current wiring away from sensitive analog modules.

I/O Module Status Indicators

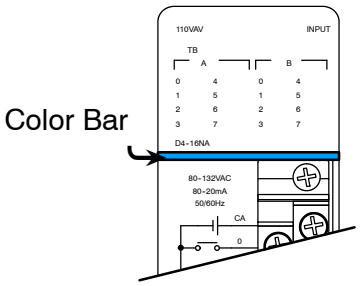
The diagram below shows the status indicator location for common I/O modules.



Color Coding of I/O Modules

The DL405 family of I/O modules have a color-coded stripe on the front bezel to help identify whether the module type is input, output, or special module. The color code meaning is listed below:

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

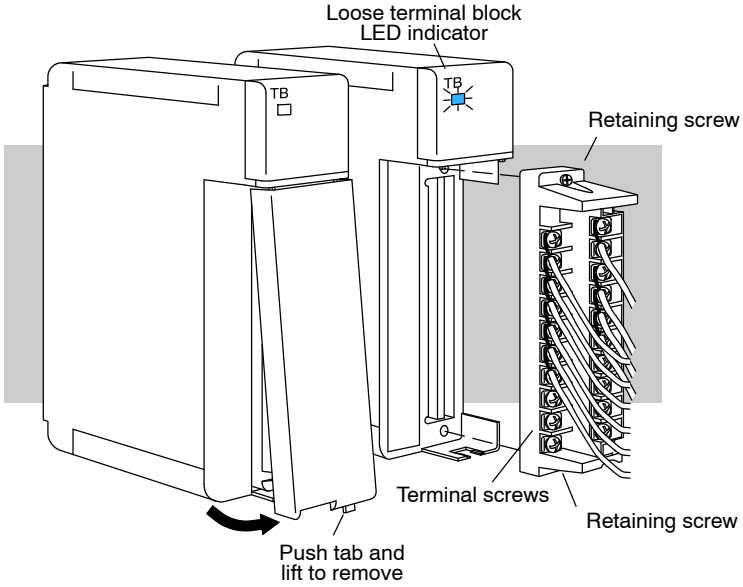


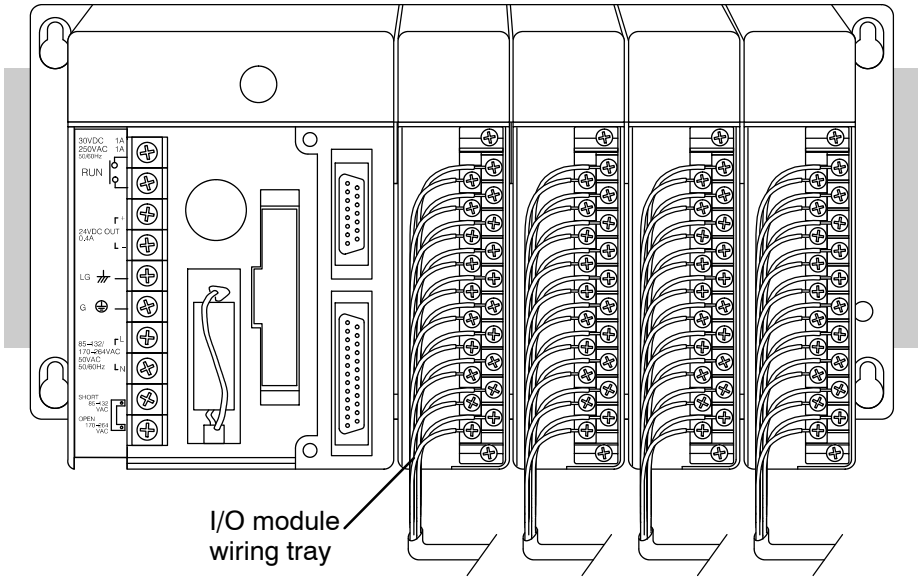
Wiring a Module with a Terminal Block

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 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.

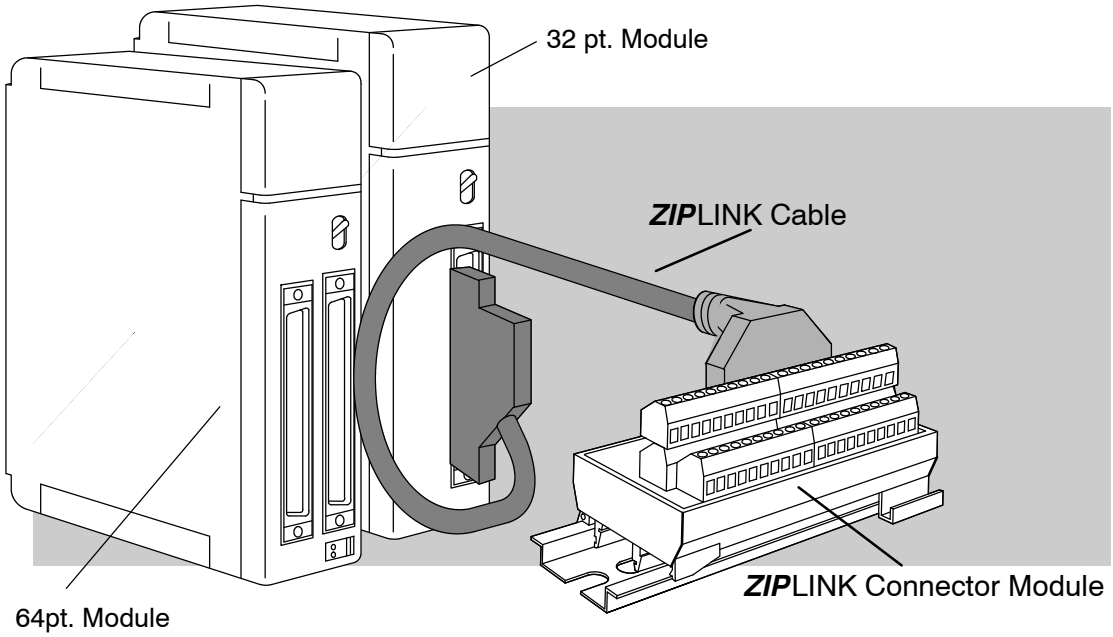
WARNING: For some modules, field device power may still be present on the terminal block even though the PLC system is turned off. To minimize the risk of electrical shock, disconnect all field device power *before* you remove the connector.





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. There are several types of connection methods available to choose from. A ***ZIP***Link connection system is shown in the figure below. Refer to the next section for complete information on ribbon and solder type connectors and accessories. Another option is to use the D4-IOCBL-1, a 3m prewired solder connector and cable with pigtail.



The ***ZIP***Link system offers "plug and play" capability, eliminating the need for traditional wiring. Simply plug one end of the ***ZIP***Link cable into a 32 or 64 point I/O module and the other end into a ***ZIP***link Connector Module. Refer to the Connection Systems section in the catalog for a complete list of cable and connector part numbers.

Part Numbers for Module Connectors

Both types of connectors are available from AutomationDirect.

AutomationDirect Part Numbers

- D4-IO3264R — Ribbon cable connectors, 2 in a pack. Can be used on either 32 point or 64 point modules.
- D4-IO3264S — Solder type connector, 2 in a pack. Can be used on either 32 point or 64 point modules.

Ribbon Cable

The chart below lists cables which can be used to connect the terminal block with a 32 I/O module. They have 40 conductors and .050" pitch PVC stranded ribbon cable.

Description/Type	Vendor	Part Number
Gray / 26 AWG	3M	3801 / 40
Gray / 26 AWG	Belden	9L260 40
Gray / 28 AWG	Belden	9L280 40
Gray / 28 AWG	DuPont	76825-040
Gray / 28 AWG	AMP	499116-5
Color coded / 26 AWG	3M	3811 / 40
Color coded / 28 AWG	Belden	9R280 40
Color coded / 28 AWG	DuPont	76177-040

Ribbon Cable Connectors

These ribbon cable connectors are for attaching the ribbon cable to the terminal block. They are all .100" x .100" 2 x20 female ribbon connectors with a center bump.

Description/Type	Vendor	Part Number
Connector Strain Relief	3M 3M	3417-7640 3448-3040
Connector Strain Relief	3M 3M	3417-7640 3448-3040
Connector (pre-assembled) Strain Relief	3M 3M	89140-0103-T0 3448-89140
Connector (with strain relief)	Thomas & Betts	622-4041
Connector (pre-assembled) Strain Relief	AMP AMP	746286-9 499252-1
Connector (with strain relief)	DuPont	66902-240
Connector (with strain relief)	Molex	15-29-9940

Interface Terminal Block

Below are terminal blocks which can be used to transition a 40 conductor ribbon cable to 40 discrete field wires. The terminal block features are: 2 x 20 .100" x .100" pin center (male) connector head terminals (.2" centers) accepting 22-12 AWG, no fuses.

Description/Type	Vendor	Part Number
Panel Mount Rail Mount	Weidmuller	RI-40A /914897 RI-40A /914908
Rail Mount	Phoenix Contacts	FLKM 40 / 2281076
Special Mount (DIN rail compatible) includes ribbon connector	Augat/RDI	2M40FC

I/O Wiring Checklist

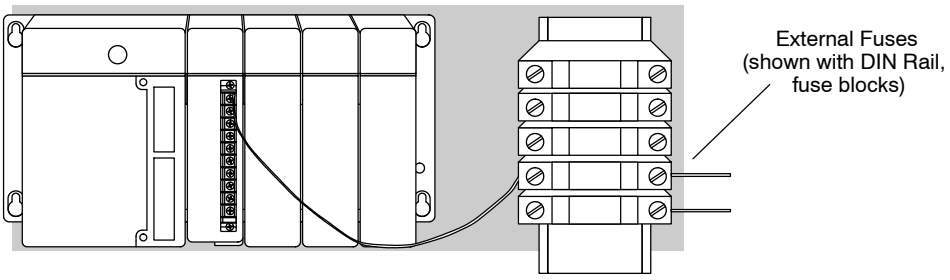
Use the following guidelines when wiring the I/O modules in your system.

1. Note the limits to the size of wire the modules can accept. The table below lists the maximum AWG for each module type. Smaller AWG is acceptable to use for each of the modules.

Module type	Suggested AWG Range	Suggested Torque
CPU	12 AWG	10.63 lb-inch (1.2 N•m)
8 point	12 AWG	7.97 lb-inch (0.9 N•m)
16 point	14 AWG	7.97 lb-inch (0.9 N•m)
32 point 64 point (connectors sold separately)	***Zip***Link: ZL-4CBL4# cable / ZL-CM40 connector block D4-IOCBL-1 (3m pigtail cable with D4-IO3264S) D4-IO3264R (ribbon type connector) D4-IO3264S (solder type connector)	

Note: 12 AWG Type TFFN or Type MTW can be used on 8pt. modules. 14 AWG Type TFFN or Type MTW can be used on 16pt. modules. 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.

2. Always use a continuous length of wire. Do not splice wires to attain a needed length.
3. Use the shortest possible wire length.
4. Where possible use wire trays for routing .
5. Avoid running wires near high energy wiring.
6. Avoid running input wiring close to output wiring where possible.
7. To minimize voltage drops when wires must run a long distance, consider using multiple wires for the return lines.
8. Where possible avoid running DC wiring in close proximity to AC wiring.
9. Avoid creating sharp bends in the wires.
10. **IMPORTANT!** To help avoid 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.



DL405 Input Module Chart

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

DL405 Input Module Type	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	✓	✓	

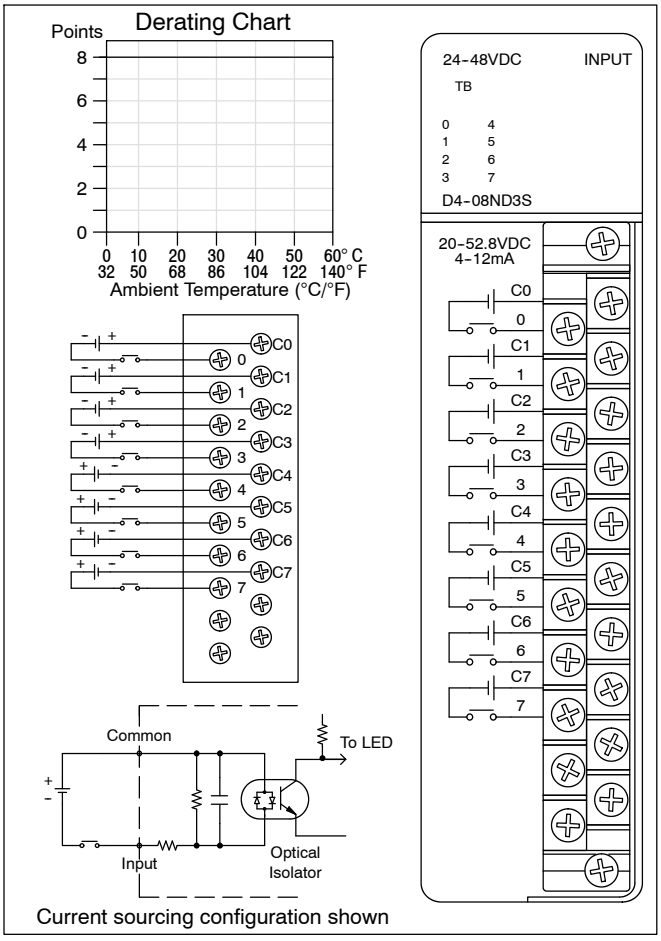
DL405 Output Module Chart

The following table lists the available DL405 output modules. Specifications begin after the input modules' specifications.

DL405 Output Module Type	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

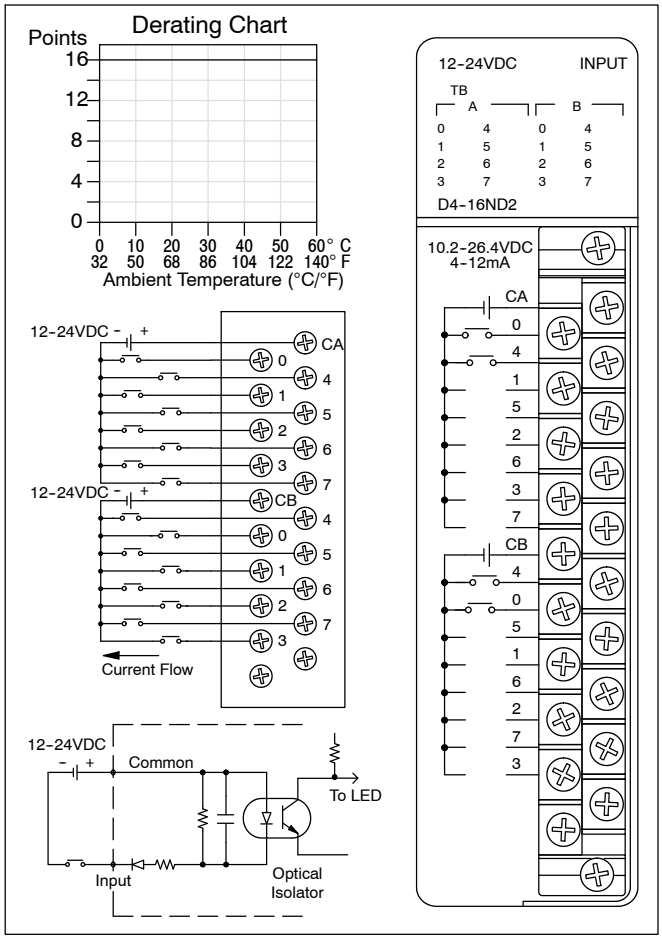
Inputs per module	8 (sink/source)
Commons per module	8 (isolated)
Input voltage range	20-52.8VDC
Peak voltage	52.8VDC
ON voltage level	>18 V
OFF voltage level	< 7V
Input impedance	4.8 K Ω
Input current @ 24 / 48 VDC	5 mA / 10 mA
Minimum ON current	3.5 mA
Maximum OFF current	1.5 mA
Base power required 5V	100 mA max
OFF to ON response	3-10 ms
ON to OFF response	3-12 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	8.8 oz. (250 g)



Current sourcing configuration shown

D4-16ND2 DC Input

Inputs per module	16 (current sourcing)
Commons per module	2 (isolated)
Input voltage range	10.2-26.4VDC
Peak voltage	26.4VDC
ON voltage level	> 9.5VDC
OFF voltage level	< 4.0 VDC
Input impedance	3.2 KΩ @ 12VDC 2.9 KΩ @24VDC
Input current @ 12 / 24VDC	3.8 mA / 8.3 mA
Minimum ON current	3.5 mA
Maximum OFF current	1.5 mA
Base power required 5V	150 mA max
OFF to ON response	1-7 ms (2.3 typical)
ON to OFF response	2-12 ms (4.6 typical)
Terminal type	Removable
Status indicators	Logic Side
Weight	8.8 oz. (250 g)

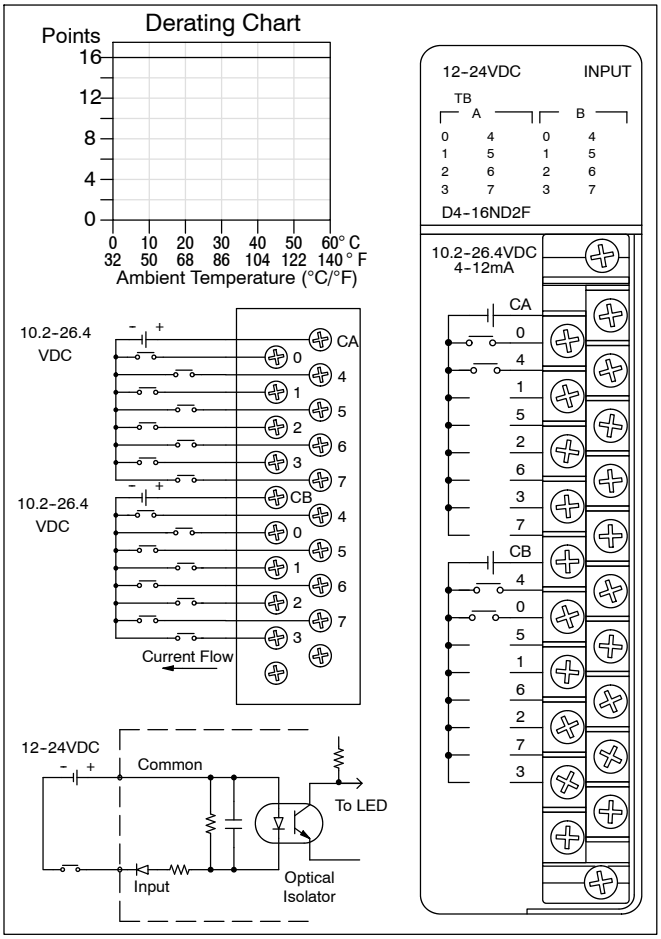


D4-16ND2F DC Input

Inputs per module	16 (current sourcing)
Commons per module	2 (isolated)
Input voltage range	10.2-26.4VDC
Peak voltage	26.4VDC
ON voltage level	> 9.5VDC
OFF voltage level	< 4.0VDC
Input impedance	3.2 K Ω @ 12VDC 2.9 K Ω @ 24VDC
Input current @ 12 / 24 VDC	3.8 mA / 8.3 mA
Minimum ON current	3.5 mA
Maximum OFF current	1.5 mA
Base power required 5V	150 mA max
OFF to ON response	1 ms
ON to OFF response	1 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	8.8 oz. (250 g)

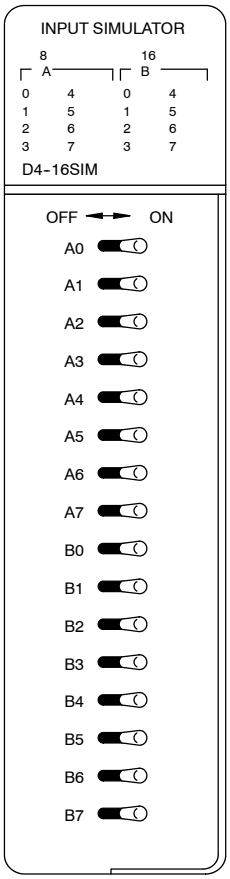
D4-16SIM Input Simulator

Inputs per module	8 or 16, selectable
Base power required 5V	150 mA Max
Terminal type	None
Status indicators	Logic Side
Weight	8.8 oz. (250 g)



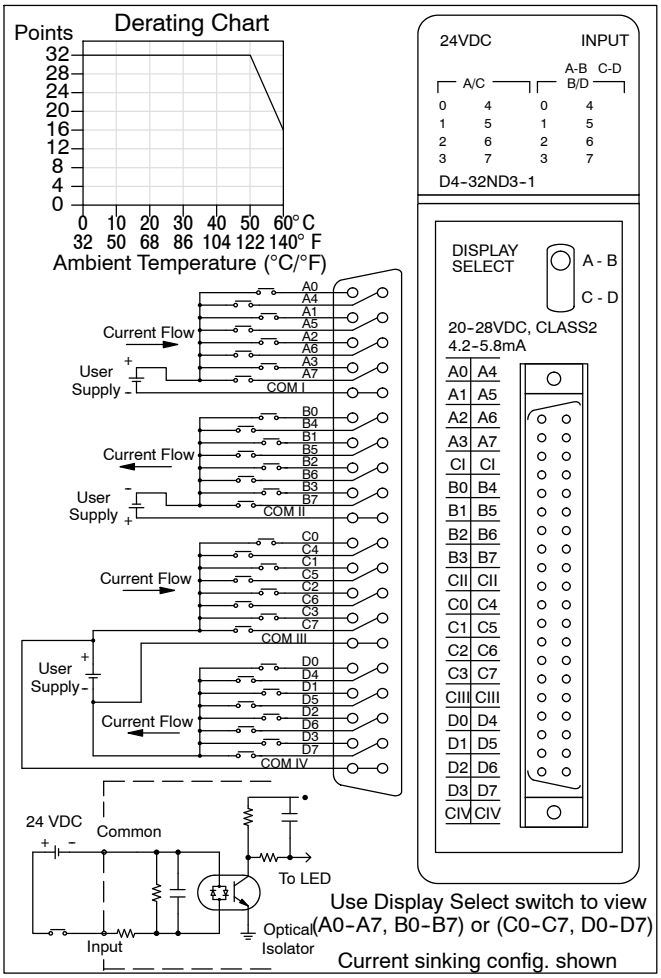
8 or 16 input point selection switch is located on the back of the module

Switch position is indicated by the LEDs above the input switches



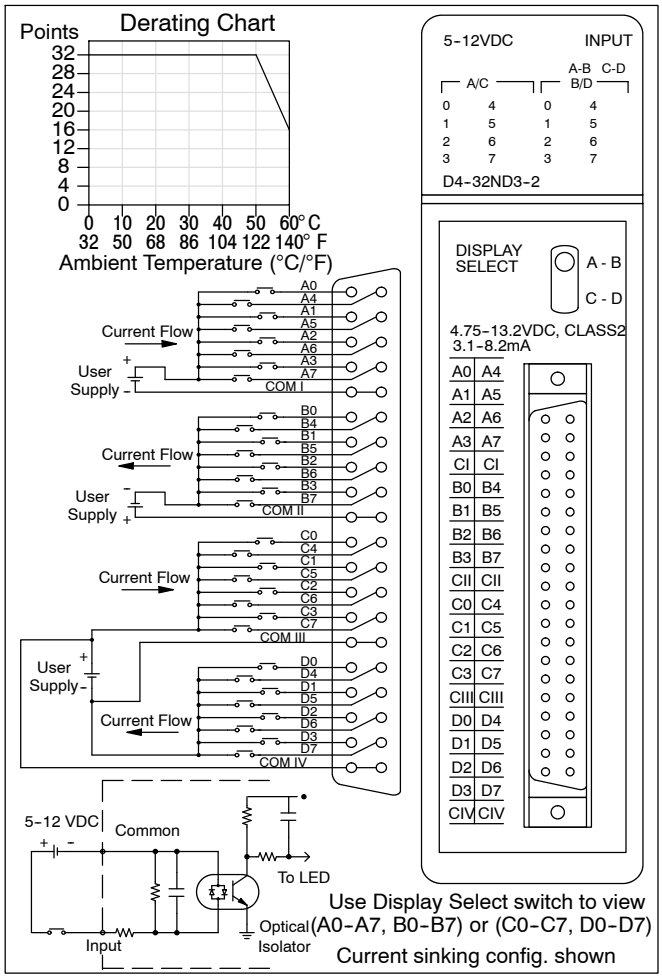
D4-32ND3-1, 24VDC Input

Inputs per module	32 (sink/source)
Commons per module	4 (isolated)
Input voltage range	20-28VDC
Peak voltage	30VDC
ON voltage level	> 19V
OFF voltage level	< 10 V
Input impedance	4.8 K Ω
Input current	5 mA
Minimum ON current	3.5 mA
Maximum OFF current	1.6 mA
Base power required 5V	150 mA max
OFF to ON response	2-10 ms
ON to OFF response	2-10 ms
Terminal type	Removable, 40 pin conn.
Status indicators	Logic Side
Weight	6.6 oz. (190 g)



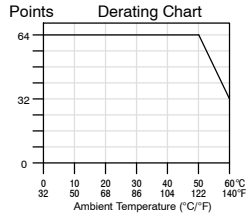
D4-32ND3-2 5-12VDC Input

Inputs per module	32 (sink/source)
Commons per module	4 (isolated)
Input voltage range	4.75-13.2VDC (TTL, CMOS)
Peak voltage	15VDC
ON voltage level	> 4 V (use pullup R for TTL in)
OFF voltage level	< 2 V
Input impedance	2 KΩ @ 5V,1.6 KΩ @ 12V
Input current	3.1 mA @ 5V, 7.5 mA @ 12V
Minimum ON current	1.8 mA
Maximum OFF current	0.8 mA
Base power required 5V	150 mA max
OFF to ON response	1-4 ms
ON to OFF response	1-4 ms
Terminal type	Removable, 40 pin conn.
Status indicators	Logic Side
Weight	6.6 oz. (190 g)

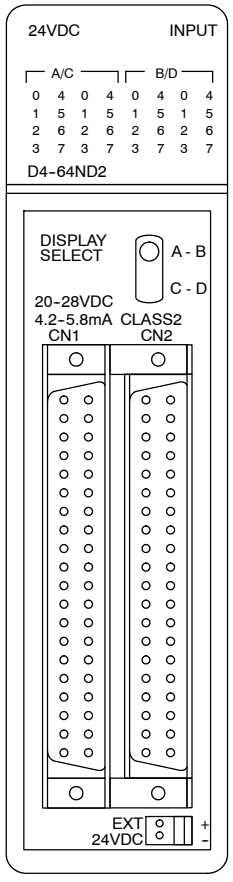


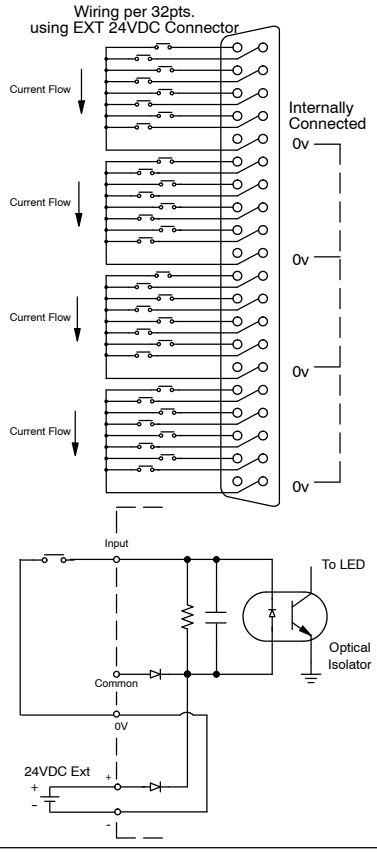
D4-64ND2, 24 VDC Input Module

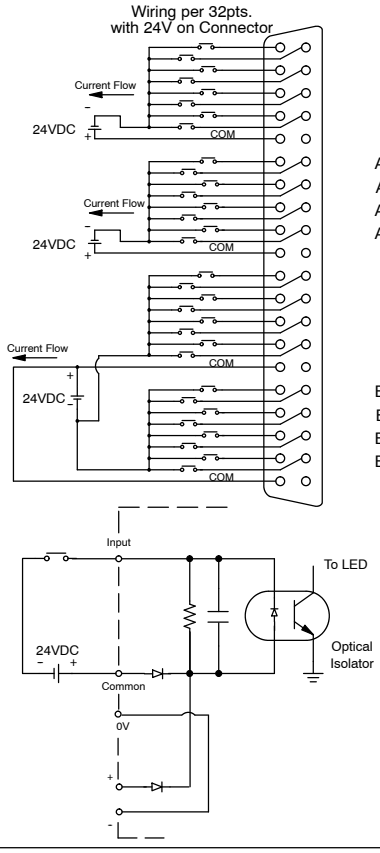
Module Location	CPU base only *		Base power required 5V	300 mA max
Inputs per module	64 (current sourcing)		External power required (optional)	24VDC ±10%, 320mA max
Commons per module	8 (isolated)		OFF to ON response	2.5 ms (typical)
Input voltage range	20 – 28 VDC		ON to OFF response	5.0 ms (typical)
Peak voltage	30 VDC		Terminal type	2, Removable 40 pin connectors (sold separately)
ON voltage level	> 20 V			
OFF voltage level	< 13 V		Status indicators	Logic Side
Input impedance	4.8 K Ω		Weight	7.8 oz. (220 g)
Input current	5.0 mA @ 24 VDC			
Minimum ON current	3.6 mA			
Maximum OFF current	2.6 mA			

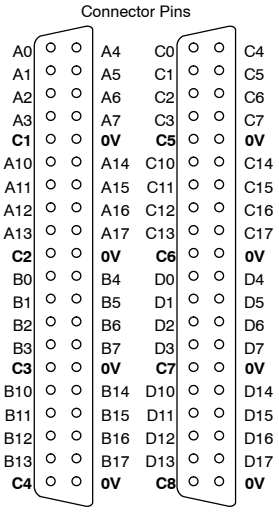


Since there are only 32 LEDs on the module, you can only display the status for 32 points at one time. In the A - B position the status of the first group of 32 input points (A0–A17, B0–B17) are displayed (connector 1). In the C - D position the status of the second group of 32 input points (C0–C17,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









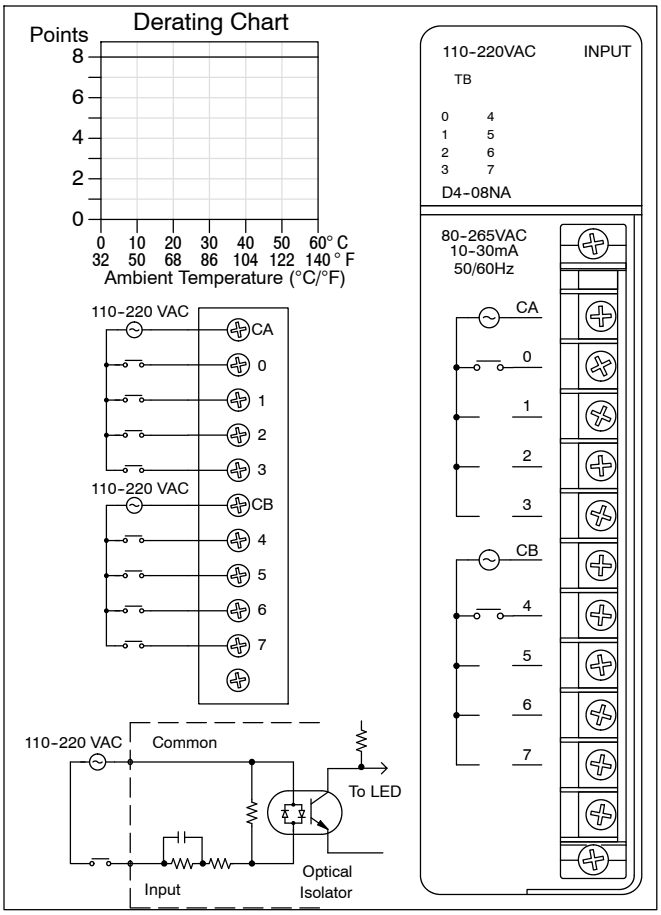
***** Module location – this module placement is restricted to the local base on DL430/DL440 systems. It may also be placed in expansion bases in DL450 systems that are using the new (-1) bases.

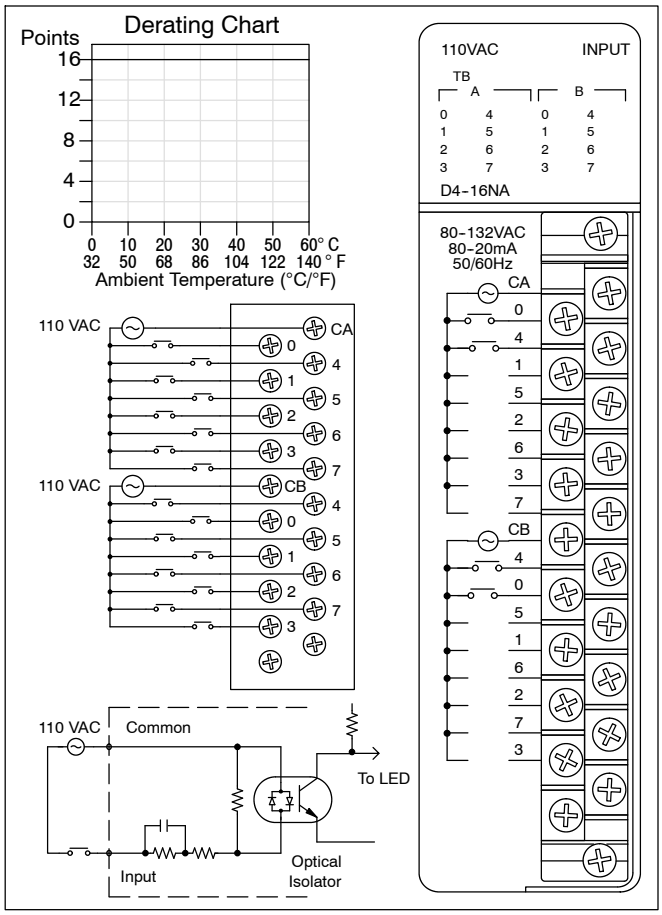
D4-08NA 110-220VAC Input

Inputs per module	8
Commons per module	2 (isolated)
Input voltage range	80-265VAC
Peak voltage	265VAC
AC frequency	47-63 Hz
ON voltage level	> 70V
OFF voltage level	< 30 V
Input impedance	12 K Ω
Input current	8.5 mA @100VAC 20 mA @ 230VAC
Minimum ON current	5 mA
Maximum OFF current	2 mA
Base power required 5V	100 mA max
OFF to ON response	5-30 ms
ON to OFF response	10-50 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	8.4 oz. (240 g)

D4-16NA 110VAC Input

Inputs per module	16
Commons per module	2 (isolated)
Input voltage range	80-132VAC
Peak voltage	132VAC
AC frequency	47-63 Hz
ON voltage level	> 70V
OFF voltage level	< 20 V
Input impedance	8 K Ω
Input current	14.5 mA @120VAC
Minimum ON current	7 mA
Maximum OFF current	2 mA
Base power required 5V	150 mA max
OFF to ON response	5-30 ms
ON to OFF response	10-50 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	9.5 oz. (270 g)



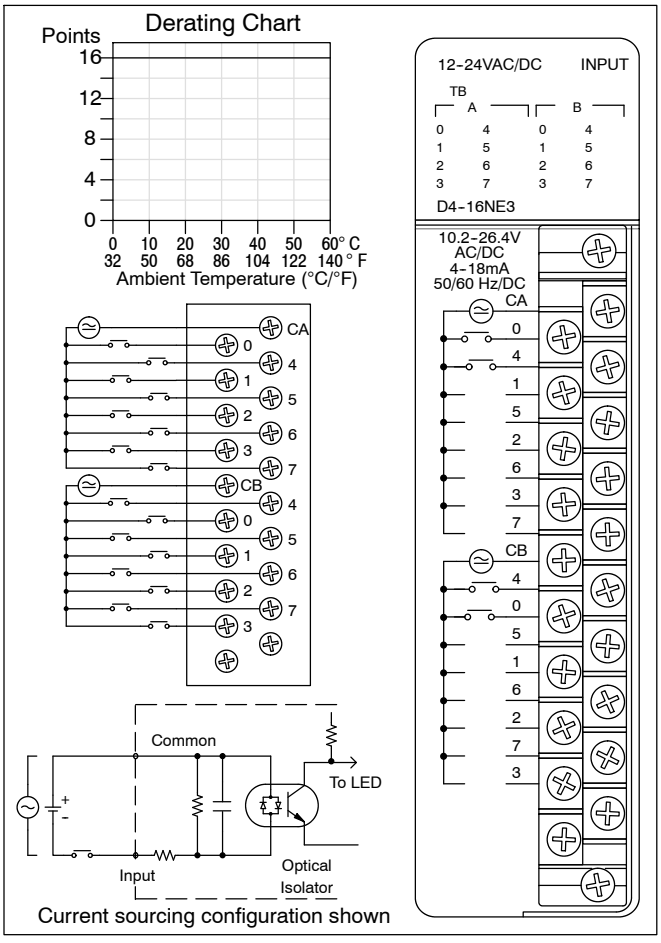


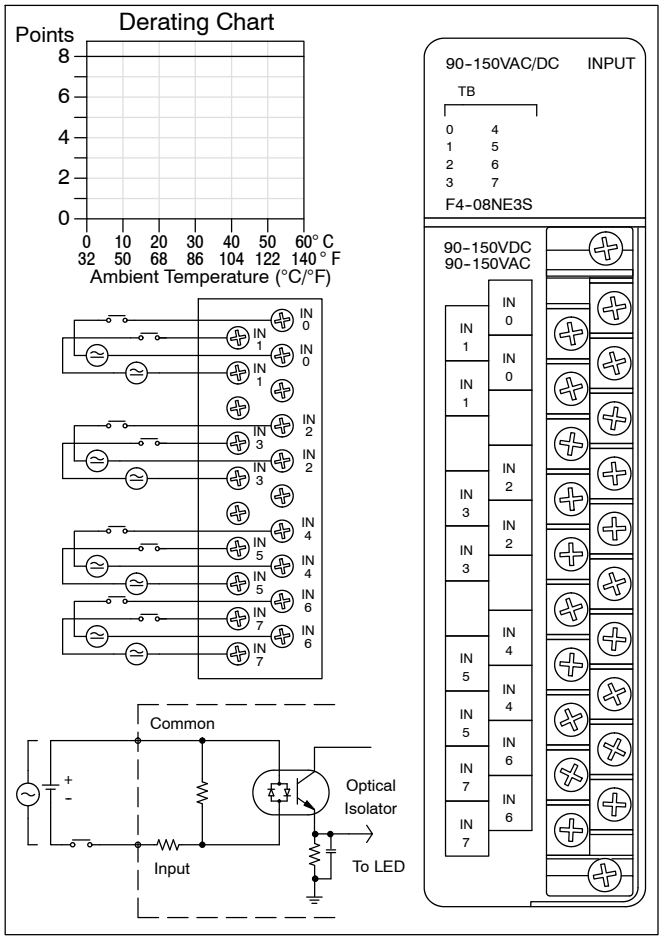
D4-16NE3 12-24VAC/DC Input

Inputs per module	16 (sink/source)
Commons per module	2 (isolated)
Input voltage range	10.2–26.4VAC/VDC
Peak voltage	37.5VAC/VDC
AC frequency	47–63 Hz
ON voltage level	> 9.5V
OFF voltage level	< 3.0V
Input impedance @ 12V/24V	3.2 K Ω / 2.9 K Ω
Input current @ 12V / 24V	3.8 mA / 8.3 mA
Minimum ON current	4 mA
Maximum OFF current	1.5 mA
Base power required 5V	150 mA max
OFF to ON response	5–40 ms
ON to OFF response	10–50 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	8.8 oz. (250 g)

F4-08NE3S 90-150VAC/DC In

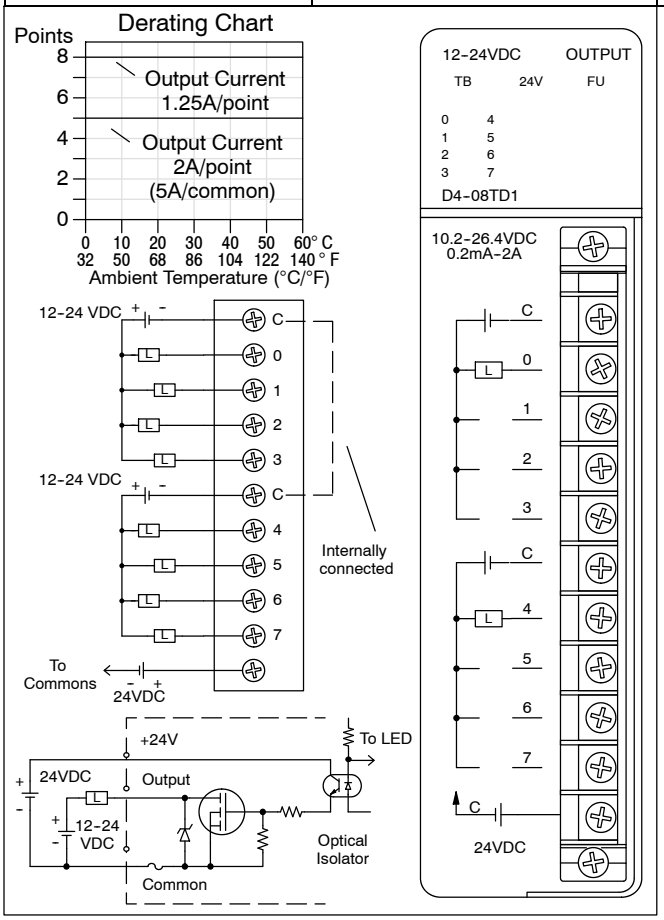
Inputs per module	8 (sink/source)
Commons per module	8 (isolated)
Input voltage range	90–150 VAC/VDC
Peak voltage	350 peak < 1ms
AC frequency	47–63 Hz
ON voltage level	> 90 VDC / 75VAC
OFF voltage level	< 60 VDC / 45VAC
Input impedance	22 K Ω
Input current	5.5 mA @ 120V
Minimum ON current	4 mA
Maximum OFF current	2 mA
Base power required 5V	90 mA max
OFF to ON response	8 ms
ON to OFF response	15 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	9 oz. (256 g)





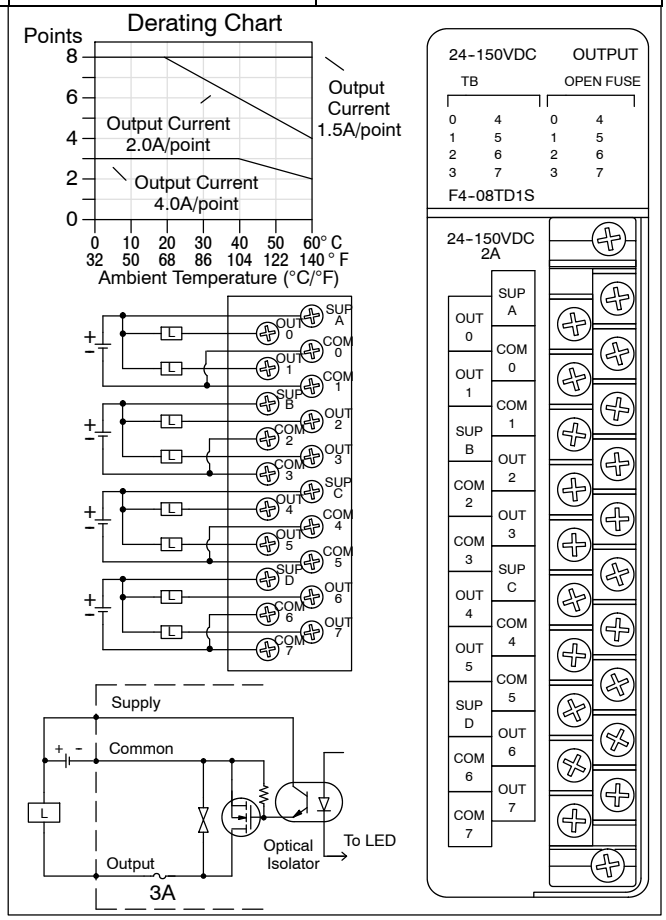
D4-08TD1 12-24 VDC Output

Outputs per module	8 (current sinking)
Commons per module	2 internally connected
Operating voltage	10.2-26.4VDC
Output type	NMOS FET (open drain)
Peak voltage	40VDC
ON voltage drop	0.5VDC @ 2A, 0.2 VDC @1A
Max current (resistive)	2A / point, 5A / common
Max leakage current	0.1mA @ 40VDC
Max inrush current	12A for 10 ms, 6A for 100 ms
Minimum load	0.2mA
Base power required 5V	150mA max
External DC required	24VDC ±10%@35 mA
OFF to ON response	1 ms
ON to OFF response	1 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	8.4 oz. (240 g)
Fuses (non-replaceable)	1 (7A) per common



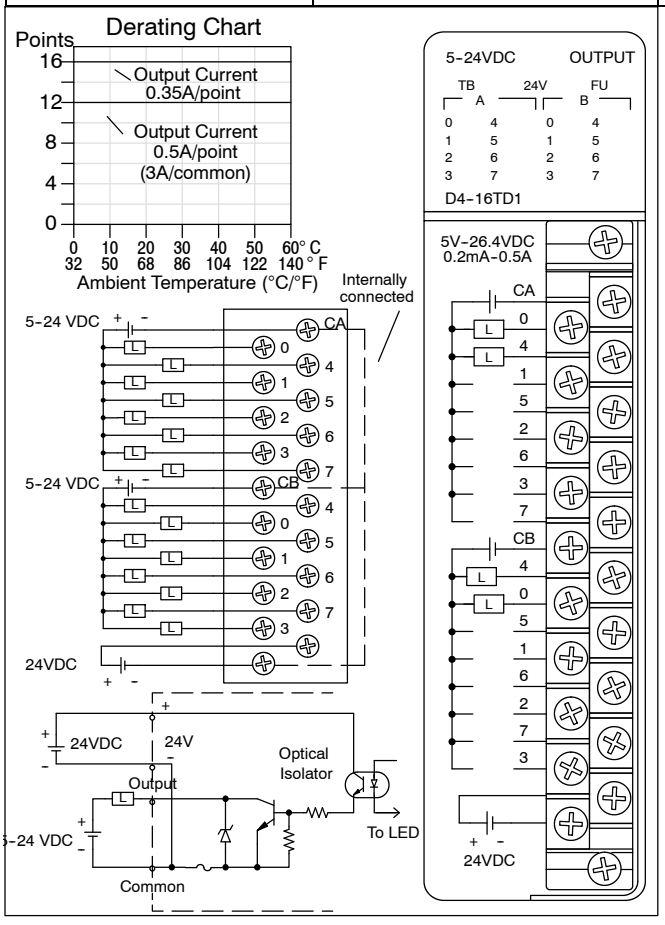
F4-08TD1S 24-150 VDC Isolated Out

Outputs per module	8 (current sinking)
Commons per module	4 (isolated)
Operating voltage	24-150VDC
Output type	MOS FET
Peak voltage	200 VDC, <1mS
ON voltage drop	1VDC @ 2A
Max current	2A / point, 4A / common
Max leakage current	5 μA
Max inrush current	30A /1ms, 6A / 10ms, 3A / 100ms
Minimum load	N/A
Base power required 5V	295 mA max
External DC required	None
OFF to ON response	25 μs
ON to OFF response	25 μs
Terminal type	Removable
Status indicators	Logic Side
Weight	10 oz. (282 g)
Fuses (non-replaceable)	1 (3A) per output



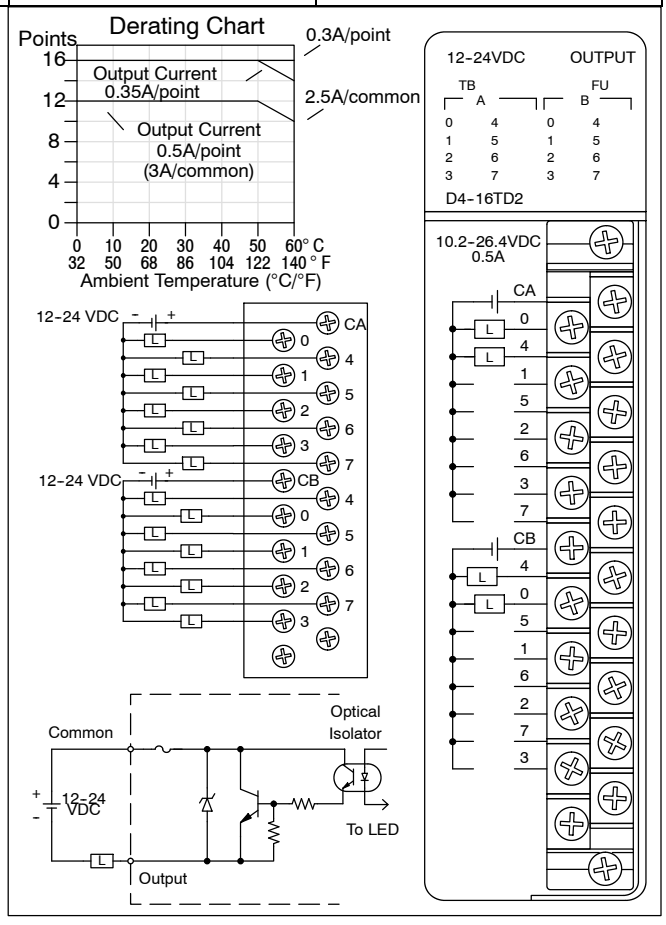
D4-16TD1 5-24 VDC Output

Outputs per module	16 (current sinking)
Commons per module	2 internally connected
Operating voltage / peak	4.5–26.4VDC, 40 VDC Peak
Output type	NPN Open collector
ON voltage drop	0.5V @ 0.5A, 0.2V @ 0.1A
Max current (resistive)	0.5A / point, 3A / common
Max leakage current	0.1mA @ 40VDC
Max inrush current	2A for 10 ms, 1A for 100 ms
Minimum load	0.2mA
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	Removable
Status indicators	Logic Side
Weight	9.5 oz. (270 g)
Fuses (non-replaceable)	1 (5A) per common



D4-16TD2, 12–24 VDC Output

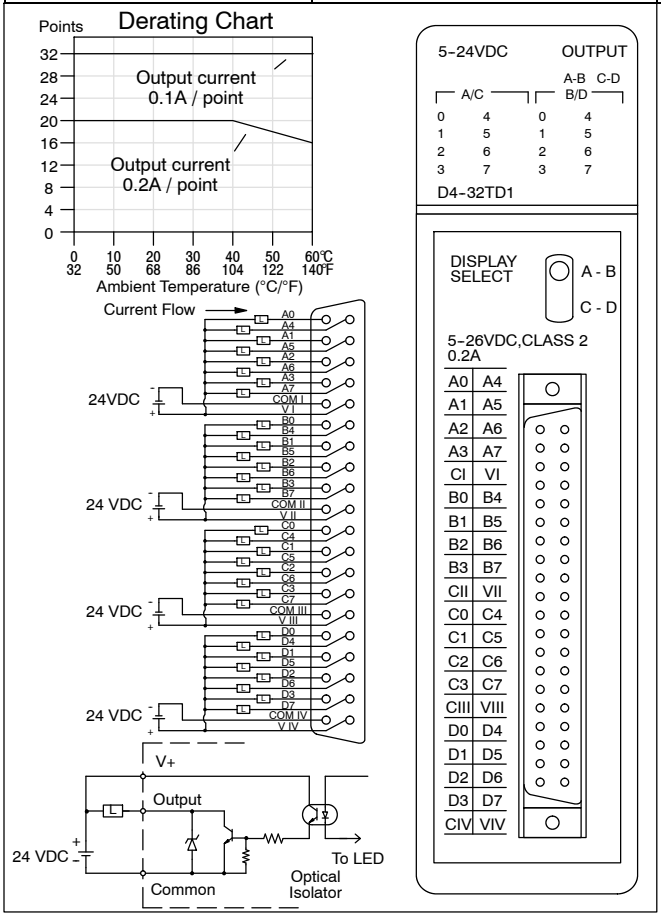
Outputs per module	16 (current sourcing)
Commons per module	2 (isolated)
Operating voltage / peak	10.2–26.4 VDC, 40 VDC Peak
Output type	NPN Emitter Follower
ON voltage drop	1.5 VDC @ 0.5A
Max current (resistive)	0.5A / point, 3A / common @ 50° C, 2.5A /common @ 60°C
Max leakage current	0.1mA @ 40 VDC
Max inrush current	2A for 10 ms, 1A for 100 ms
Minimum load	0.2mA
Base power required 5V	400mA max
External DC required	None
OFF to ON response	1 ms
ON to OFF response	1 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	9.8 oz. (280 g)
Fuses (non-replaceable)	1 (5A) per common



See Errata Sheet at the beginning of this file. The D4-32TD1-1 operates on reverse logic. Voltage is present when Output is off; no voltage is present when output is on.

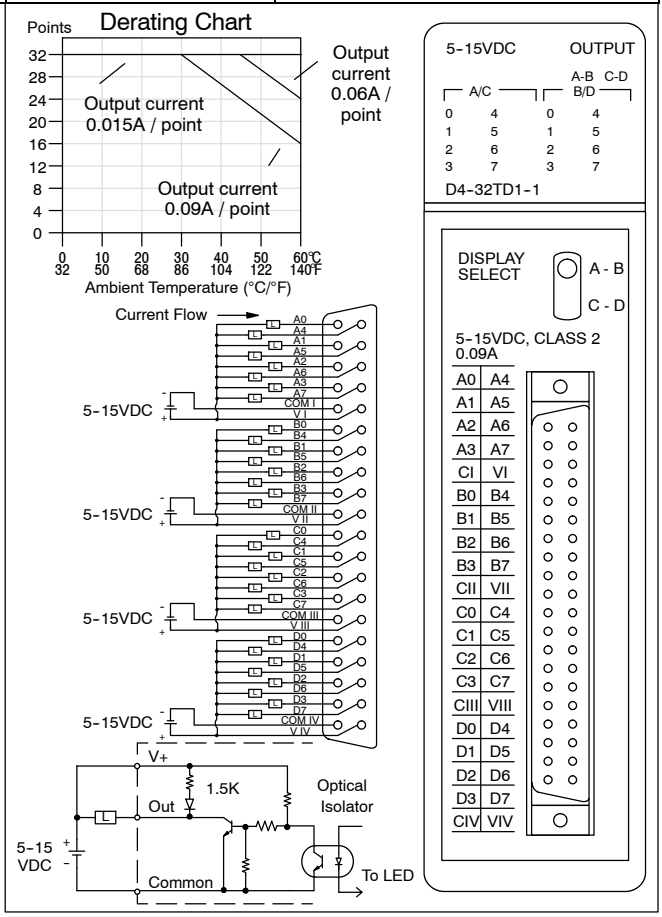
D4-32TD1, 5-24VDC Output

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	36 VDC
ON voltage drop	0.6 VDC @ 0.2A
Max current (resistive)	0.2A / point, 1.6A / common
Max leakage current	0.1mA @ 36 VDC
Max inrush current	1A for 10 ms, 0.5A for 100 ms
Minimum load	0.1mA
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	Removable
Status indicators	Logic Side
Weight	6.7 oz. (190 g)
Fuses	None



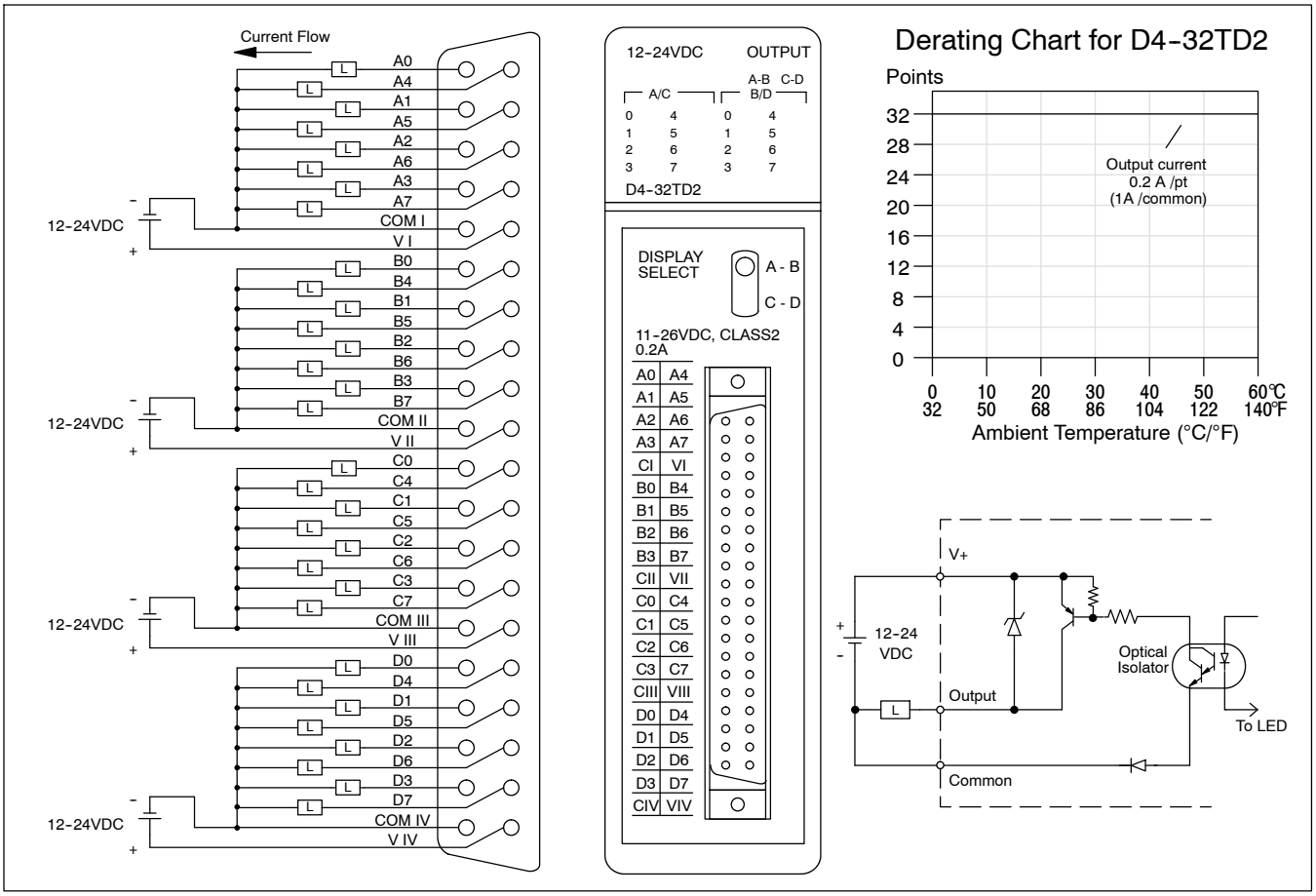
D4-32TD1-1, 5-15VDC Output

Outputs per module	32 (current sinking)
Commons per module	4 (isolated)
Operating voltage	5-15 VDC
Output type	NPN Open Collector (w / pullup)
Peak voltage	16.5 VDC
ON voltage drop	0.4 VDC @ 0.1A
Max current (resistive)	0.09A/pt, 0.72A/com, 2.88A/ mod.
Max leakage current	0.01mA @ 16.5 VDC
Max inrush current	0.5A for 10ms, 0.2A for 100ms
Minimum load	0.15mA
Base power req., 5V	250mA max
External DC required	5-15VDC ±10%, 150mA max
OFF to ON response	0.1 ms
ON to OFF response	0.1 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	6.7 oz. (190 g)
Fuses	None



D4-32TD2, 12-24 VDC Output Module

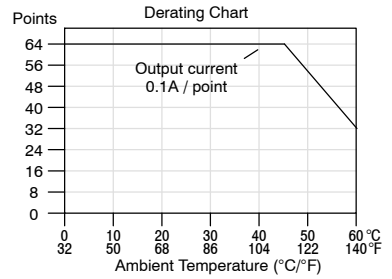
Outputs per module	32 (current sourcing)		External DC required	10.8-26.4VDC 1A / common including load
Commons per module	4 (isolated)			
Operating voltage	10.8-26.4 VDC			
Output type	PNP Open Collector		OFF to ON response	< 0.2 ms
Peak voltage	30 VDC		ON to OFF response	< 0.2 ms
ON voltage drop	0.6 VDC @ 0.2A		Terminal type	Removable
Max current (resistive)	0.2A / point 1.0A / common 4.0A / module		Status indicators	Logic Side
			Weight	6.7 oz. (190 g)
Max leakage current	0.01mA @ 26.4 VDC		Fuses	None
Max inrush current	500 mA for 10 ms			
Minimum load	0.2mA			
Base power required 5V	350mA max			



Only 16 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 16 output points (A0-A7, B0-B7) is displayed.
In the C - D position the status of the second group of 16 output points (C0-C7, D0-D7) is displayed.

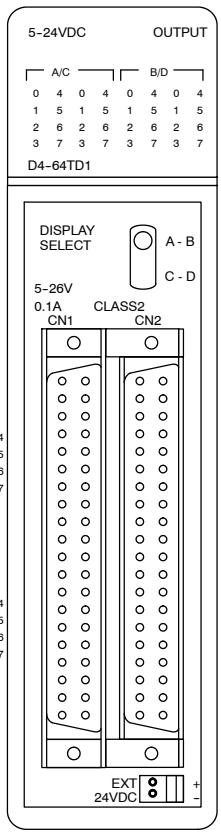
D4-64TD1, TTL/CMOS/5-24 VDC Output Module

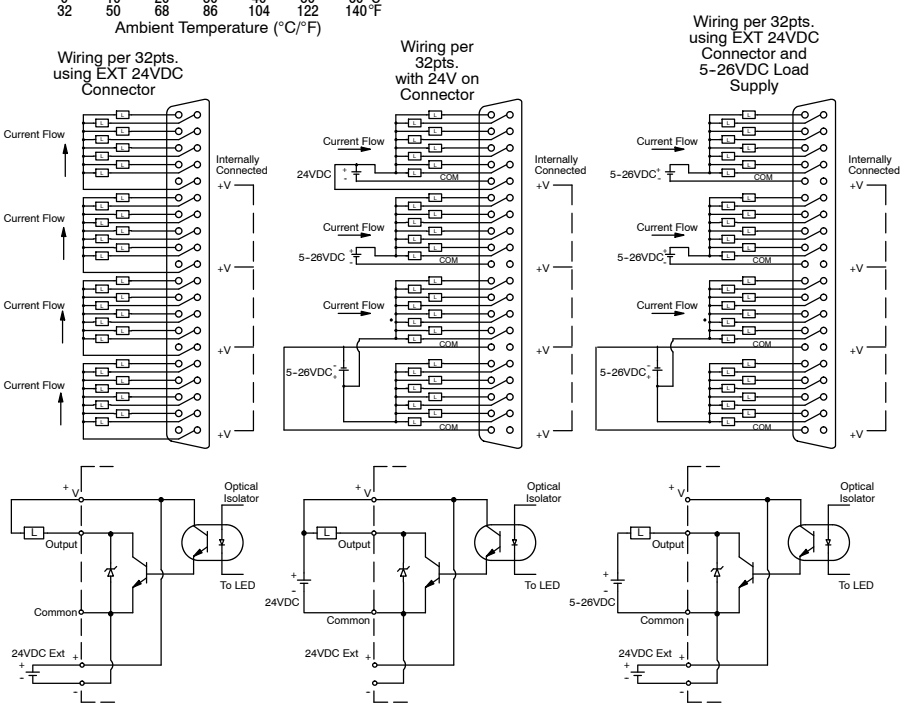
Module Location	CPU base only *		Minimum load	0.1mA
Outputs per module	64 (current sinking)		Base power required 5V	800mA max
Commons per module	8 (non-isolated)			
Operating voltage	4.75-26.5 VDC			
Output type	NPN Open Collector		External DC required	24VDC ±10%, (800mA + 50mA per common) 7.0A total max
Peak voltage	36 VDC		OFF to ON response	< 0.1 ms
ON voltage drop	0.6 VDC @ 0.1A		ON to OFF response	< 0.2 ms
Max current (resistive)	0.1A / point 1.0A / common 8.0A / module		Terminal type	2, Removable 40-pin connectors (sold sep.)
			Status indicators	Logic Side
Max leakage current	0.01mA @ 36 VDC		Weight	7.4 oz. (210 g)
Max inrush current	1A for 1 ms 700mA for 100 ms		Fuses	None



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).

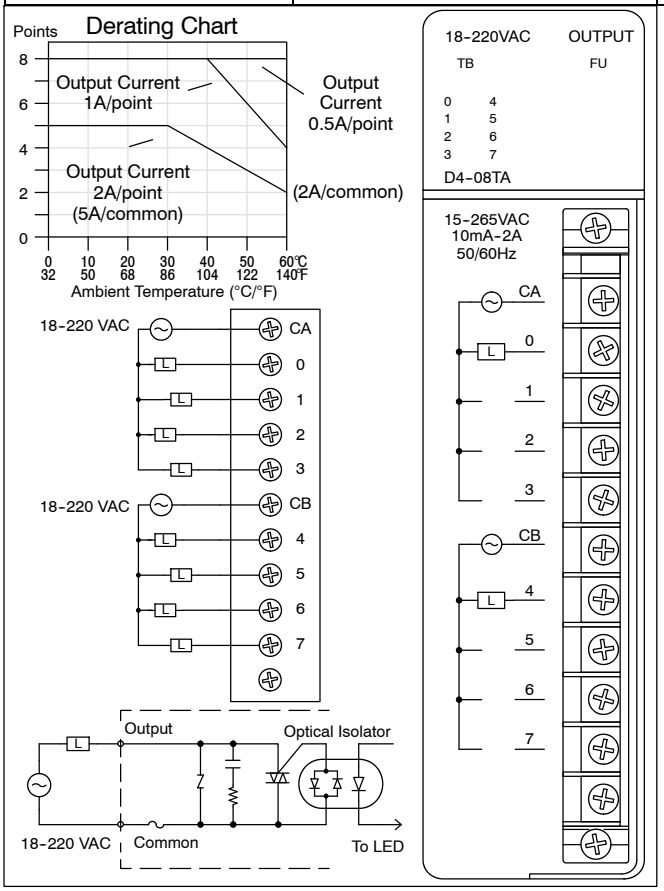
* Module location - this module placement is restricted to the local base on DL430/DL440 systems. It may also be placed in expansion bases in DL450 systems that are using the new (-1) bases.





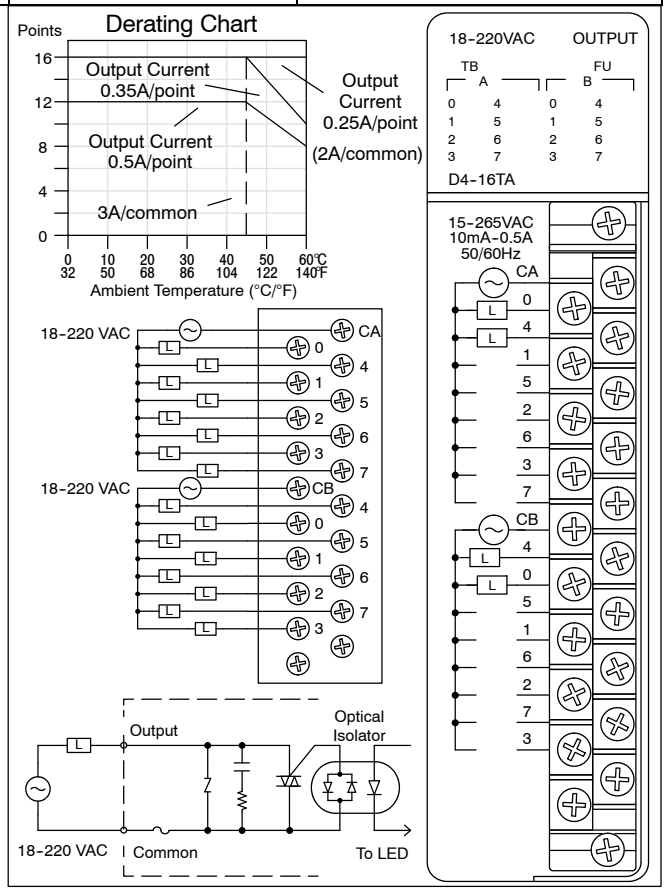
D4-08TA, 18-220VAC Output

Outputs per module	8
Commons per module	2 (isolated)
Operating voltage	15-265VAC
Output type	SSR (triac)
Peak voltage	265VAC
AC frequency	47-63 Hz
ON voltage drop	1.5VAC @ 2A
Max current	2A / point, 5A / com. @ 30°C 2A / common @ 60 °C
Max leakage current	5mA @ 265VAC
Max inrush current	30A for 10 ms, 10A for 100 ms
Minimum load	10 mA
Base power required 5V	250 mA max
OFF to ON response	1 ms
ON to OFF response	1 ms +1/2 AC cycle
Terminal type	Removable
Status indicators	Logic Side
Weight	11.6 oz. (330 g)
Fuses (non-replaceable)	1 (8A) per common



D4-16TA, 18-220VAC 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 @ 0.5A
Max current	0.5A / pt, 3A / common @ 45 °C 2A / common @ 60 °C
Max leakage current	4mA @ 265VAC
Max inrush current	15A for 10 ms, 10A for 100 ms
Minimum load	10 mA
Base power required 5V	450 mA max
OFF to ON response	1 ms
ON to OFF response	1 ms +1/2 AC cycle
Terminal type	Removable
Status indicators	Logic Side
Weight	12.2 oz. (350 g)
Fuses (non-replaceable)	1 (5A) per common

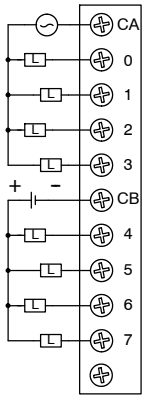


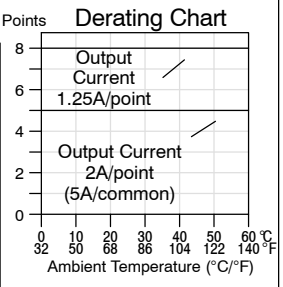
D4-08TR, Relay Output

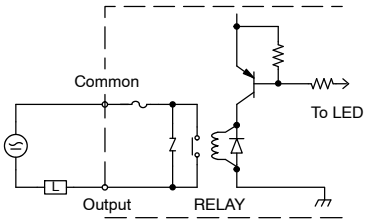
Outputs per module	8 relays
Commons per module	2 (isolated)
Operating voltage	5-30VDC / 5-250VAC
Output type	Form A (SPST-NO)
Peak voltage	30VDC / 256VAC
AC frequency	47-63 Hz
Max current (resistive)	2A / point, 5A / common
Max leakage current	0.1mA @ 265VAC
Max inrush current	2A
Minimum load	5mA
Base power required 5V	550mA max
External DC required	None
OFF to ON response	12 ms
ON to OFF response	12 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	9.1 oz. (260 g)
Fuses (non-replaceable)	1 (8A) per common

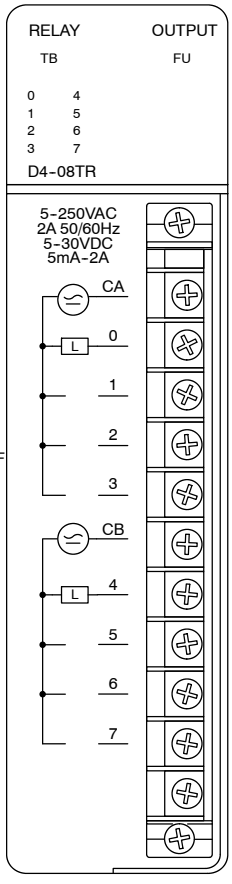
Typical Relay Life (Operations)

Maximum Resistive or Inductive Inrush Load Current	Operating Voltage		
	30VDC	125VAC	250VAC
2A resistive	100K	300K	200K
2A inductive	100K	80K	60K
0.5A resistive	800K	1M	800K
0.5A inductive	300K	300K	200K







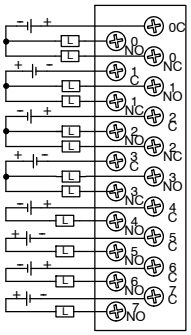


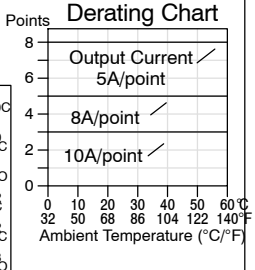
F4-08TRS-1, Relay Output

Outputs per module	8 relays
Commons per module	8 (isolated)
Operating voltage:12-30VDC, 12-125VAC, 125-250VAC*	
Output type: 4, Form C (SPDT), 4, Form A (SPST-NO)	
Peak voltage	30VDC / 250VAC @10A
AC frequency	47-63 Hz
Max current (resistive)	10A / point, 40A / module
Max leakage current	0.1mA @ 265VAC
Max inrush current	10A
Minimum load	100mA @12 VDC
Base power required 5V	575mA max
External DC required	None
OFF to ON response	7 ms
ON to OFF response	9 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	13.2 oz. (374 g)
Fuses (non-replaceable)	1 (10A/125V) per common

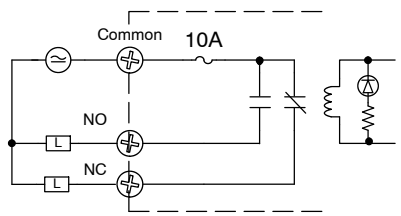
Typical Relay Life (Operations)

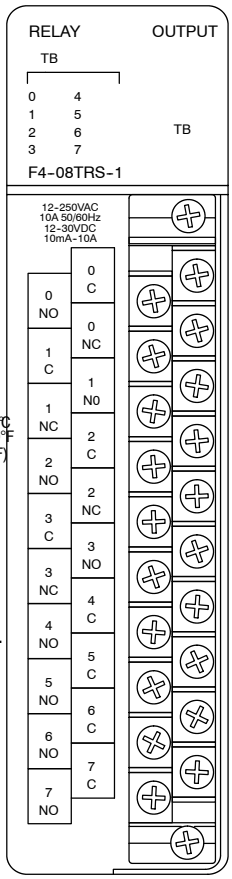
Maximum Resistive or Inductive Inrush Load Current	Operating Voltage		
	30VDC	125VAC	250VAC
1/4 HP		25K	
10.0A	50K	50K	
5.0A	200K	100K	
3.0A	325K	125K	50K
0.5A	>50M		





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.



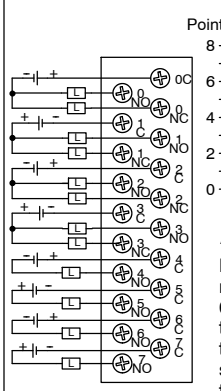


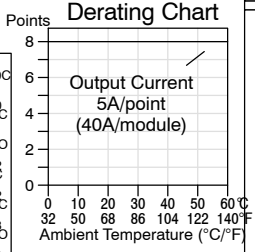
F4-08TRS-2, Relay Output

Outputs per module	8 relays
Commons per module	8 (isolated)
Operating voltage	12-30VDC, 12-250VAC
Output type: 4 Form C (SPDT), 4 Form A (SPST-NO)	
Peak voltage	30VDC / 250VAC @5A
AC frequency	47-63 Hz
Max current (resistive)	5A / point, 40A / module
Max inrush current	10A
Minimum load	100mA @12 VDC
Base power required 5V	575mA max
External DC required	None
OFF to ON response	7 ms
ON to OFF response	9 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	13.8 oz. (390 g)
Fuses, (user replaceable)	1 (10A, 250V) per common 19379-K-10A Wickman

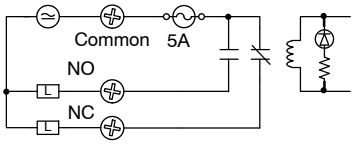
Typical Relay Life (Operations)

Maximum Resistive or Inductive Inrush Load Current	Operating Voltage		
	28VDC	120VAC	240VAC
5.0A	200K	100K	
3.0A	325K	125K	50K
.05A	>50M		

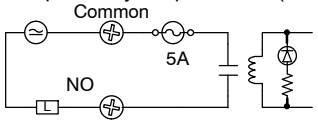




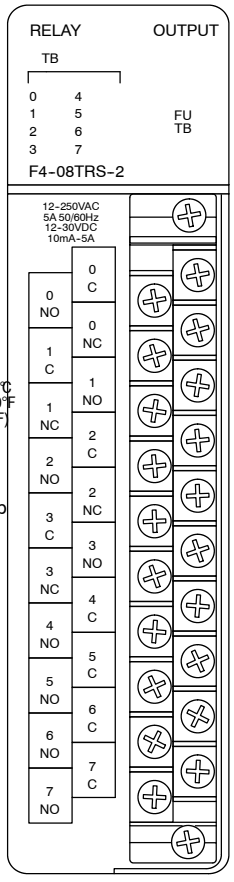
Maximum DC voltage rating is 120 VDC @ 0.5A, 30,000 cycles typical. Motor starters up to and including NEMA size 3 can be used with this module.



Sample Relay Output Circuit (1 of 4)



Sample Relay Output Circuit (1 of 4)

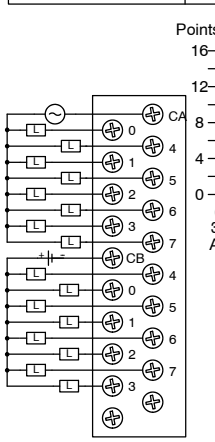


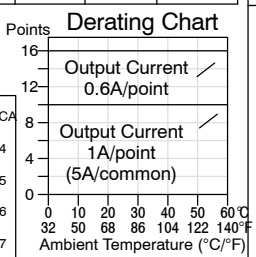
D4-16TR, Relay Output

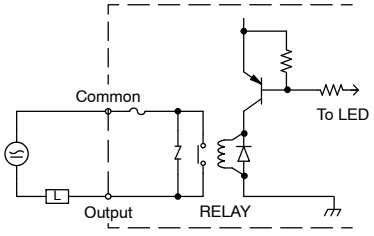
Outputs per module	16 relays
Commons per module	2 (isolated)
Operating voltage	5-30VDC / 5-250VAC
Output type	Form A (SPST-NO)
Peak voltage	30VDC / 256VAC
AC frequency	47-63 Hz
Max current (resistive)	1A / point, 5A / common
Max leakage current	0.1mA @ 265VAC
Max inrush current	4A
Minimum load	5mA
Base power required 5V	1000mA max
External DC required	None
OFF to ON response	10 ms
ON to OFF response	10 ms
Terminal type	Removable
Status indicators	Logic Side
Weight	10.9 oz. (310 g)
Fuses (non-replaceable)	1 (8A) per common

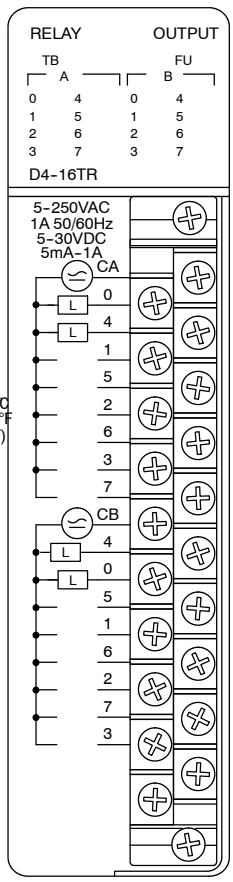
Typical Relay Life (Operations)

Maximum Resistive or Inductive Inrush Load Current	Operating Voltage		
	30VDC	125VAC	250VAC
1A resistive	>1M	500K	300K
1A inductive	400K	200K	100K
0.5A resistive	>2M	800K	500K
0.5A inductive	>1M	300K	200K









Glossary of Specification Terms

Inputs or Outputs Per Module	Indicates number of electrical input or output points per module and designates current sinking, current sourcing, or either.
Commons Per Module	Number of electrical commons per module. A common is a connection to an input or output module which is shared by multiple I/O circuits. It is ususally in the return path to the power supply of the I/O circuit.
Input Voltage Range	The operating voltage range of an input circuit, measured from an input point to its common terminal, when the input is ON.
Output Voltage Range	The output voltage range of an output circuit, measured from an output point to its common terminal, when the output is OFF.
Peak Voltage	Maximum voltage allowed for an input or output circuit for a short duration.
AC Frequency	AC modules are designed to operate within a specific frequency range.
ON Voltage Level	The minimum voltage level at which an input point will turn ON.
OFF Voltage Level	The maximum voltage level at which an input point will turn OFF.
Input Impedance	The electrical resistance measured between an input point and its common point. Since this resistance is non-linear, it may be listed for various input currents.
Input Current	Typical operating current for an active (ON) input.
Minimum ON Current	The minimum current for the input circuit to operate reliably in the ON state.
Maximum OFF Current	The maximum current for the input circuit to operate reliably in the OFF state.
Minimum Load	The minimum load current required for an output circuit to operate properly.
External DC Required	Some output modules require external power for the output circuitry.
On Voltage Drop	Sometimes called “saturation voltage”, it is the voltage measured from an output point to its common terminal when the output is ON, at max. load.
Maximum Leakage Current	The maximum current a connected maximum load will receive when the output point is OFF.
Maximum Inrush Current	The maximum current used by a load for a short duration upon an OFF to ON transition of a output point. It is greater than the normal ON state current and is characteristic of inductive loads in AC circuits.
Base Power Required	The +5VDC power from the base required to operate the module. Be sure to observe the base power budget calculations.
OFF to ON Response	The time the module requires to process an OFF to ON state transition.
ON to OFF Response	The time the module requires to process an ON to OFF state transition.
Status Indicators	The LEDs that indicate the ON/OFF status of an input or output point. These LEDs are electrically located on the logic (CPU) side of the I/O interface circuit.
Terminal Type	Indicates whether the module’s connector is removable or non-removable.
Weight	Indicates the weight of the module. See Appendix E for a list of the weights for the various DL405 components.
Fuses	Protective device for an output circuit, which stops current flow when current exceeds the fuse rating current. It may be replaceable or non-replaceable, or located externally or internally.

3

CPU Specifications and Operation

In This Chapter. . . .

— Overview
— CPU General Specifications
— CPU Electrical Specifications
— CPU Hardware Features
— Using Battery Backup
— Selecting the Program Storage Media
— CPU Setup
— CPU Operation
— I/O Response Time
— CPU Scan Time Considerations
— PLC Numbering Systems
— Memory Map
— DL405 Aliases
— X Input/Y Output Bit Map
— Control Relay Bit Map
— Timer and Counter Status Bit Maps
— Remote I/O Bit Map
— Stage™ Control / Status Bit Map

Overview

The CPU is the heart of the control system. 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 DL430, DL440, and DL450 are all modular CPUs which are installed in either 4, 6, or 8 slot bases. All I/O modules in the DL405 family will work with either CPU. The DL405 CPUs offer a wide range of processing power and program instructions. All offer RLL and Stage program instructions (See Chapter 5 for instruction definitions). All DL405 CPUs have extensive internal diagnostics that can be monitored from the application program or from an operator interface.

The three standard CPU types accept either 110VAC or 220 VAC for power input. The DL440 CPU is available in two additional DC versions: the DL440DC-1 uses 24 VDC, and the DL440DC-2 uses 125 VDC.

DL430 CPU Features

The DL430 has 6.5K of program memory comprised of 3.5K of ladder memory and 3K of V-memory (data registers). It has 113 instructions available for program development, and supports a maximum of 640 points of local and local expansion I/O and 512 points of remote I/O.

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

The DL430 provides two built-in communications ports. The first has a RS232C interface and the other has a RS232C/RS422 interface. This allows for a point-to-point connection on the first port and the option of either a multidrop network connection (such as ***Direct***NET) or a point-to-point connection on the other port.

DL440 CPU Features

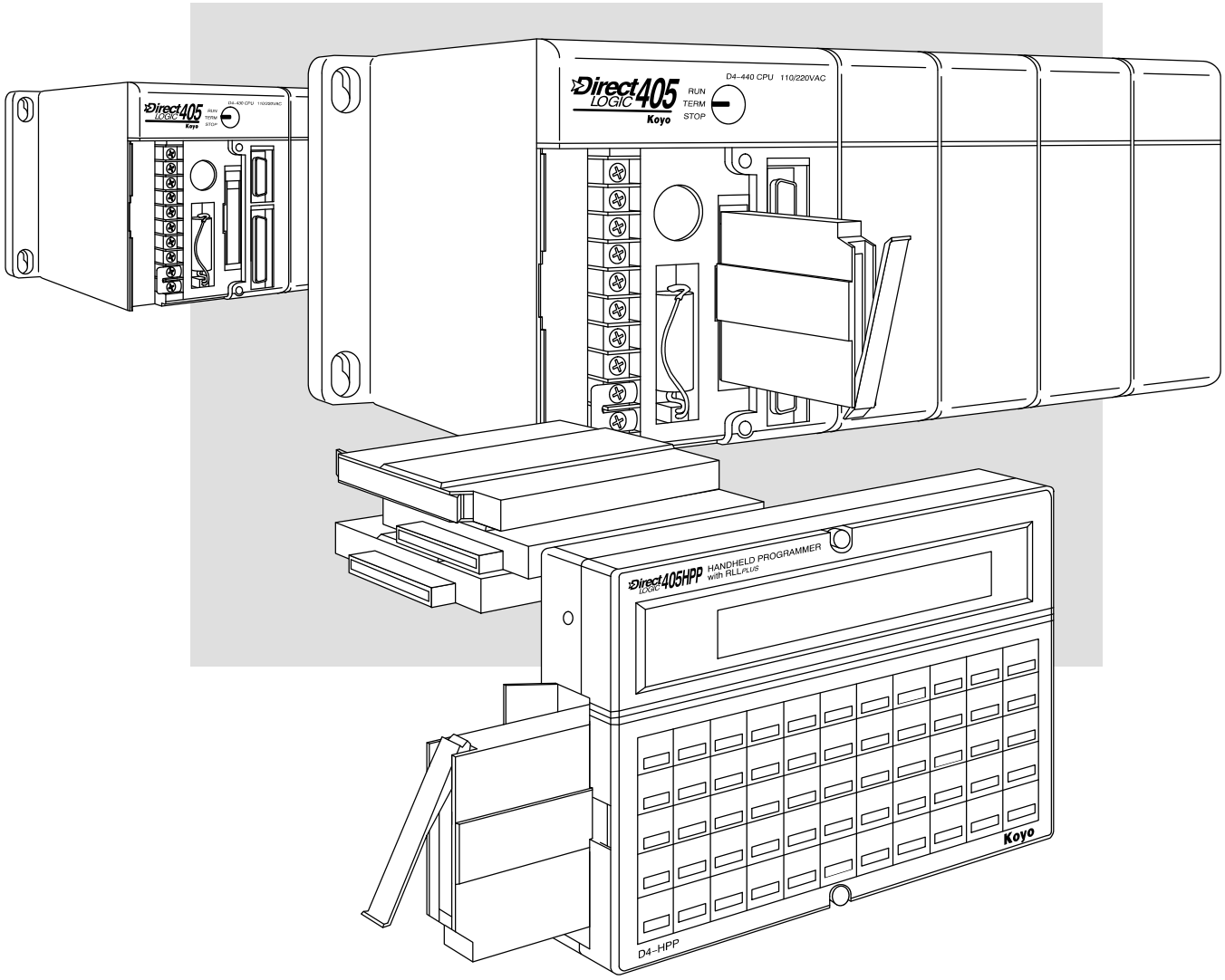
The DL440 includes all the DL430 features, plus more I/O points, more program instructions, and greater memory space with plug-in memory cartridges. It has a maximum of 22.5K of program memory comprised of 15.5K of ladder memory and 7K of V-memory (data registers). It supports a maximum of 640 points of local and local expansion I/O and 1024 points of remote I/O. Its two communications ports operate identically to the DL430's ports.

The DL440 has 170 instructions. The additional 57 instructions to the DL430 instruction set allow for more sophisticated program development through the use of subroutines, additional instructions that support double word manipulation, enhanced stack operations, diagnostic messaging, and ASCII/Hex data formatting.

DL450 CPU Features

The new DL450 offers all the DL440 features, plus more I/O points, program instructions, and two additional (4 total) communications ports. It has a maximum of 30.8K of program memory comprised of 15.5K of ladder memory and 15.3K of V-memory (data registers). It supports a maximum of 2048 points of local and local expansion I/O, and 1536 points of remote I/O. It includes an additional internal RISC–based microprocessor for greater processing power. The DL450 has 210 instructions. The additional 40 instructions to the DL440 instruction set include drum timers, a print function, floating point math, trigonometric functions, and PID loop control for 16 loops.

The DL450 has a total of four communications ports. The first two ports are identical to those on the DL430 and DL440. The third port has a RS-232C interface and can be configured for either N sequence or K sequence protocol. It uses a modular connector for point-to-point communications to devices such as the DV-1000 Data Access Unit. The fourth port has a RS-422 interface using either MODBUS master/slave, N sequence, or K sequence protocol. These four ports utilize three physical connectors (the bottom connector has two ports on the DL450).



CPU General Specifications

Features	DL430	DL440	DL450
Total Program memory (words)	6.5K	14.5K / 22.5K*	22.8K / 30.8K*
Ladder memory (words), built-in	3.5K	7.5K / 15.5K*	7.5K / 15.5K*
V–memory (words)	3.0K	7.0K	15.3K
Scan Time, typical (1 K boolean)	8 – 10 mS	2 – 3 mS	4 – 5 mS
Run time edit	No	Yes	Yes
Supports Override	No	No	Yes
RLL and RLL*PLUS* Programming	Yes	Yes	Yes
Handheld programmer with cassette tape interface	Yes	Yes	Yes
DirectSOFT programming for Windows™	Yes	Yes	Yes
Built-in communication ports	2 ports	2 ports	4 ports
CMOS RAM	No	w/mem. cartridge	w/mem. cartridge
UVPROM	No	w/mem. cartridge	w/mem. cartridge
EEPROM	Standard on CPU	w/mem. cartridge	w/mem. cartridge
FLASH RAM	No	No	Standard on CPU
Compatible with:			
CoProcessor™ modules	Yes	Yes	Yes
Networking modules	Yes	Yes	Yes
RS232C/RS422 Data Comm. Module	Yes	Yes	Yes
Total I/O	1152	1664	3584
Total I/O available as:			
Local I/O / Local expansion I/O / Remote I/O	640	640	4096
Remote I/O	512 max.	1024 max.	2048 max.
Remote I/O Channels	2	2	3
Local discrete input points maximum	320	320	1024
Local discrete output points maximum	320	320	1024
Local analog input channels maximum	320**	320**	512**
Local analog output channels maximum	320**	320**	512**
Maximum number of channels / masters (remote or slice) per local CPU base	2	2	2
Remote I/O Distance	3300 ft. (1000m)	3300 ft. (1000m)	3300 ft. (1000m)
Discrete I/O Module Point Density	8/16/32/64	8/16/32/64	8/16/32/64
Slots per Base	4/6/8	4/6/8	4/6/8

***** The first values represent CPUs using the 7.5K memory cartridge and the second value is for using 15.5K memory cartridges.

****** Additional Discrete and Analog I/O can be supported (within the power budget) through the use of remote I/O.

Feature	DL430	DL440	DL450
Number of instructions available (see Chapter 5 for a description of the available instructions)	113	170	210
Control relays	480	1024	2048
Special relays (system defined)	288	352	512
Stages in RLL*PLUS*	384	1024	1024
V-memory	3072 words	7168 words	15360 words
Timers	128	256	256
Counters	128	128	256
Immediate I/O	Yes	Yes	Yes
Interrupt input	8 points	16 points	16 points
Subroutines	No	Yes	Yes
For/Next Loops	No	Yes	Yes
Drum Timers	No	No	Yes
Math	Integer	Integer	Integer and Floating Point
PID Loop Control, built-in	No	No	16 loops
Time of Day Clock/Calendar	No	Yes	Yes
Internal diagnostics	Yes	Yes	Yes
Password security	No	Yes	Yes, multi-level
System and user error log	No	Yes	Yes
Battery backup	Yes	Yes	Yes

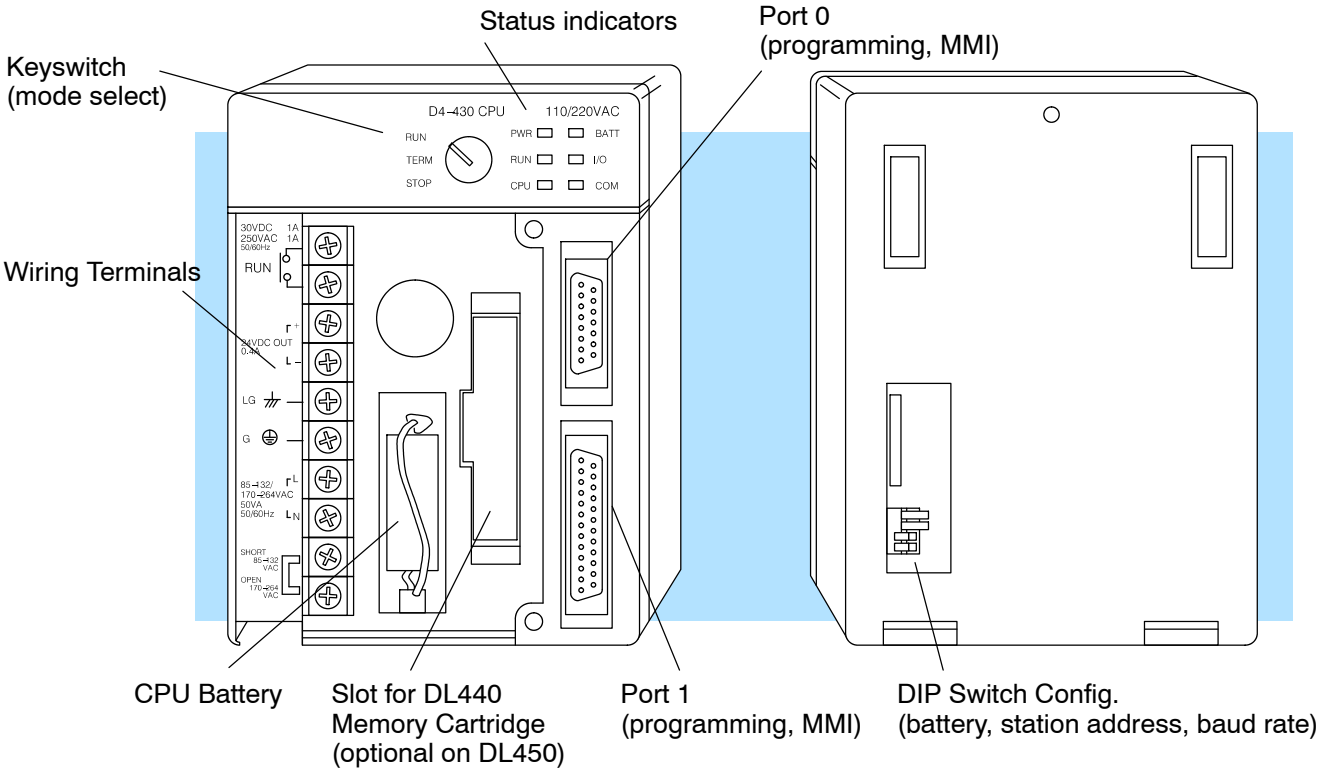
CPU Electrical Specifications

Parameter	DL430/DL440/DL450	DL440/450DC-1	DL440/450DC-2
Input Voltage, Nominal	120 VAC	24 VDC	125 VDC
Input Voltage Range	100–120 VAC and 196–240 VAC +10% –15%	20–29 VDC	100–132 VDC +10% –15%
Input Voltage Ripple	N/A	less than 10%	less than 10%
Inrush Current, maximum	20 A	10 A	20 A
Power Consumption, maximum	50 VA	38W	30 W
Voltage withstand (dielectric strength)	1 min. 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 (N/A on DL440–DC–1 and DL440–2)		
Output Current, auxiliary power supply	24 VDC @ 400 mA maximum		

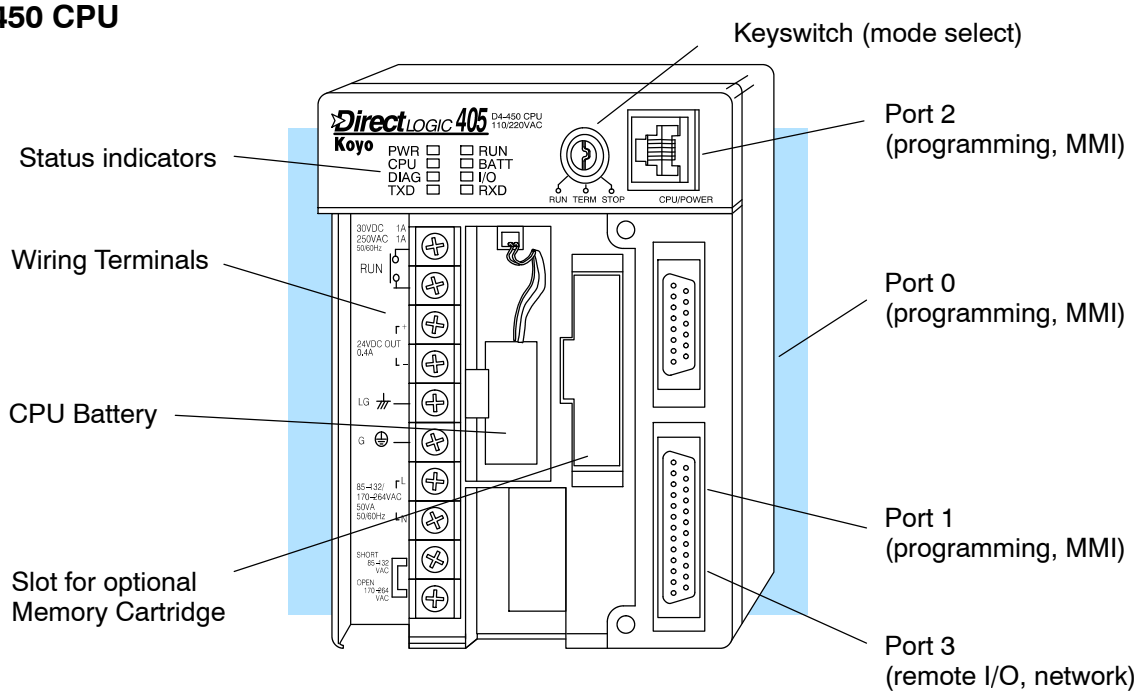
CPU Hardware Features

The following diagram shows the main external features of the DL405 CPUs.

DL430/DL440 CPUs



DL450 CPU



Keyswitch Functions

The keyswitch on the DL405 CPUs provides positions for enabling and disabling program changes in the CPU. Unless the keyswitch is in the TERM position, RUN and STOP mode changes will not be allowed by any interface device, (handheld programmer, ***Direct***SOFT programing package or operator interface). Programs may be viewed or monitored but no changes may be made. If the switch is in the TERM position and no program password is in effect, all operating modes as well as program access will be allowed through the connected programming or monitoring device.

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

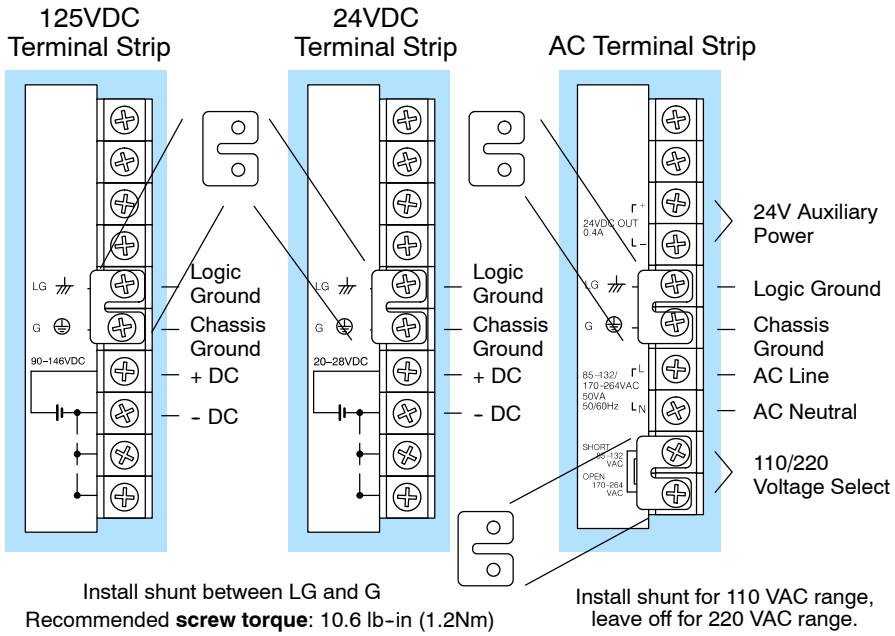
There are two ways to change the CPU mode.

1. Use the CPU key switch to select the operating mode.
2. Place the CPU key 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.

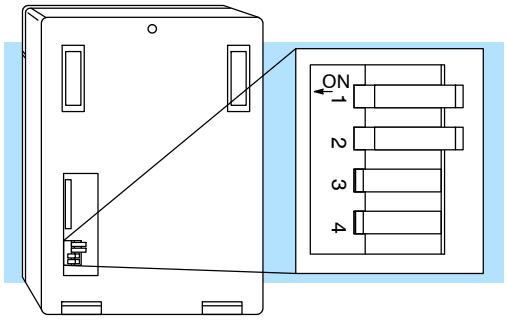
Indicator	Status	Meaning
PWR	ON	Power good
	OFF	Power failure
RUN	ON	CPU is in Run Mode
	OFF	CPU is in Stop Mode
	FLASHING	CPU is in firmware upgrade mode
CPU	ON	CPU self diagnostics error
	OFF	CPU self diagnostics good
BATT **Note:.**Refer to page 3–12	ON	CPU battery voltage is low
	OFF	CPU battery voltage is good or disabled
DIAG (DL450)	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
COM (DL430/DL440)	ON	Commuications error has occurred
	OFF	Communications OK
TXD (DL450)	ON	Data is being transmitted by the CPU
	OFF	No data is being transmitted by the CPU
RXD (DL450)	ON	Data is being received by the CPU
	OFF	No data is being received by the CPU



Setting the CPU DIP Switches

✓ ✓ ✗
430 440 450

Locate the bank of four configuration switches located on the back of DL430 and DL440 CPUs as shown in the figure below. These switches affect battery low detection, station address override and baud rate of port 1 (25 pin D connector). Use Aux Functions on the DL450 for these selections, via a programming device.



Switch 1

- ON = Battery low indicator disabled
- OFF= Battery low indicator enabled

Switch 2

- ON = Station address override is enabled (address 1)
- OFF = Station address is set by AUX function with programming device

NOTE: Setting Switch 2 on forces the station address to 1. It does not change the address set by the programming device. When Switch 2 is turned off again the address will revert back to the address stored in memory via the AUX function.

Port 1 Baud Rate	Switch 3	Switch 4
300	Off	Off
1200	Off	On
9600	On	Off
19200	On	On

NOTE: Parity, Mode and Station address for port 1 is selected by AUX functions using a programming device.

Communication Ports

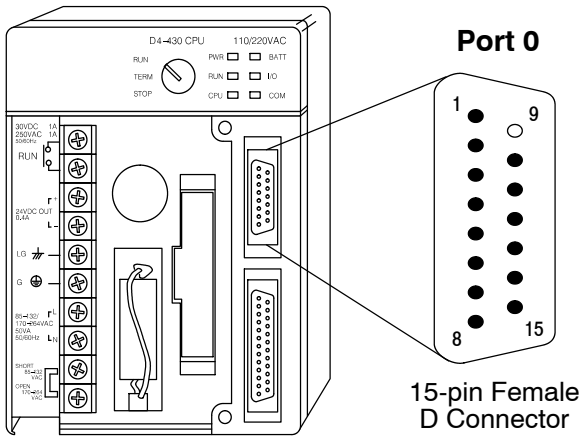
Port 0 Specifications

✓ ✓ ✓
430 440 450

DL405 CPUs provides up to four communication ports. The DL430/DL440 CPUs have two ports, while the DL450 CPU has a total of four ports.

The first port (all CPUs) is located on the 15 pin D-shell conector. It is for general programming such as ***Direct***SOFT, or operator interface connections. The D4-HPP handheld programmer can only be used on this port on the CPU. The operating parameters for Port 0 are permanently set to the values shown.

- 15 Pin female D type connector
- Protocol: K sequence
- RS232C, non-isolated, distance within 15 m (approx. 50 feet)
- 9600 baud, 8 data bits, 1 start, 1 stop bit, odd parity
- Asynchronous, Half duplex, DTE



15-pin Female D Connector

Port 0 Pin Descriptions (All CPUs)		
1	YOP	Sense connection between HPP and CPU
2	TXD	Transmit Data (RS232C)
3	RXD	Receive Data (RS232C)
4	ONLINE	Request Communication (TTL)
5	ABNO	CPU Error (TTL)
6	PRDY	CPU ready to communicate (TTL)
7	CTS	Clear to Send (RS232C)
8	YOM	Sense connection between HPP and CPU
9	-	Not Used
10	LCBL	Sense cable connection (TTL)
11	5V2	5 VDC for HPP logic
12	5V2	5 VDC for LCD backlight
13	0V	Logic ground
14	0V	Logic ground
15	0V	Logic ground

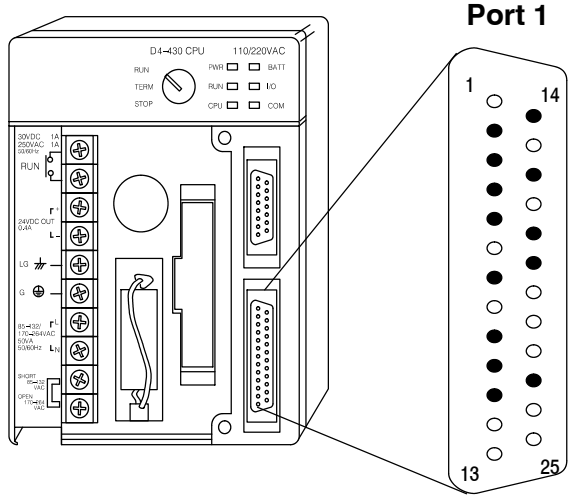
Port 1 Specifications

✓ ✓ ✓
430 440 450

Port 1 (all CPUs) is located on the 25-pin connector, and is called the “secondary comm port” for the DL430/DL440 CPUs. The secondary comm port address is stored in the memory cartridge along with the I/O configuration. It is for general programming such as ***Direct***SOFT, operator interfaces, and networking, but it cannot connect to the handheld programmer. Port 1 provides additional features such as programmable baud rate, parity, ASCII/Hex mode and network address. Its RS422 signals support multidrop networking and programming applications.

The baud rate and station address override is selected by dip switches on the rear of the DL430/DL440 CPUs. The DL450 uses Aux functions to set the same parameters (it has no DIP switches). RS232C 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 AUX (auxiliary) functions with a programming device.

- 25 Pin female D type connector
- Protocols: K-sequence, ***Direct***Net. The DL450 additionally supports Non-Sequence and MODBUS protocols. (Note: The DL430 cannot support K-sequence on ports 0 and 1 simultaneously. Use ***Direct***Net on port 1 if port 0 is used for communications).
- RS232C / RS422, Selectable address 1-90 (use Aux function)
- 300/ 600/ 1200/2400/4800/9600/19200/(38400 DL450 only) baud
- Hex / ASCII modes (use Aux function to configure)
- 8 data bits, 1 start, 1 stop bit, Odd, Even or No parity
- Asynchronous, Half duplex (use Aux function to configure), DTE



25-pin Female
D Connector

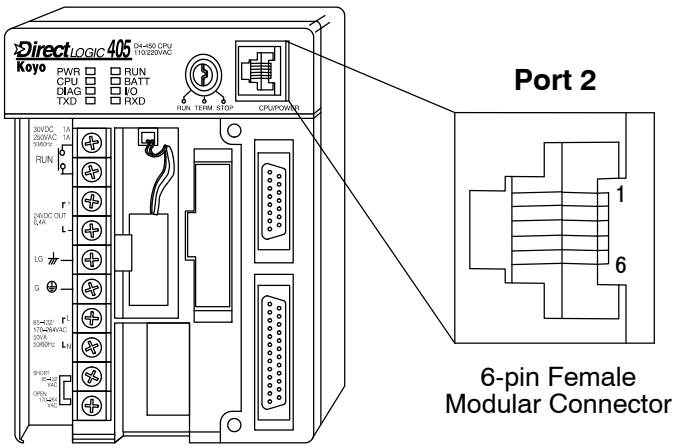
Port 1 Pin Descriptions (All CPUs)		
1	–	Not used
2	TXD	Transmit Data (RS232C)
3	RXD	Receive Data (RS232C)
4	RTS	Request to Send (RS232C)
5	CTS	Clear to Send (RS2332C)
6	–	Not used
7	SG	Signal ground (RS232C/RS422)
8	–	(port 3 on DL450)
9	RXD+	Receive Data + (RS422)
10	RXD–	Receive Data – (RS422)
11	CTS+	Clear to Send + (RS422)
12	–	(port 3 on DL450)
13	–	(port 3 on DL450)
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 on DL450)
25	–	(port 3 on DL450)

Port 2 Specifications

		✓
430	440	450

The operating parameters for Port 2 on the DL450 CPU are configurable using Aux functions on a programming device.

- 6 Pin female modular (RJ12 phone jack) type connector
- Protocols: ***Direct***Net (slave only), K sequence, Non-procedure
- RS232C, 300 / 600 / 1200 / 2400 / 4800 / 9600 / 19200 / 38400 baud
- 8 data bits, 1 start, 1 stop bit; odd, even, or no parity
- Nodes – from 1 to 90



6-pin Female
Modular Connector

Port 2 Pin Descriptions (DL450)		
1	0V	Power (-) connection (GND)
2	5V	Power (+) conection
3	RXD	Receive Data (RS232C)
4	TXD	Transmit Data (RS232C
5	5V	Power (+) conection
6	0V	Power (-) connection (GND)

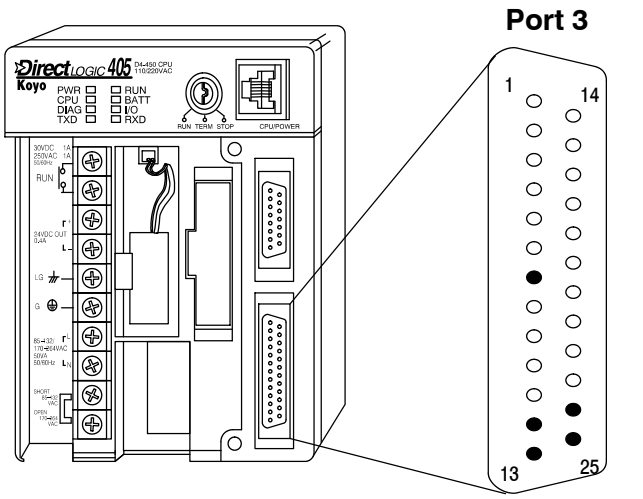
NOTE: The 5V pins are rated at 200mA maximum, primarilly for use with some operator interfaces.

Port 3 Specifications

430	440	450
✗	✗	✓

The operating parameters for Port 3 on the DL450 CPU are configurable using Aux functions on a programming device.

- 25 Pin female D type connector
- Protocols: DirectNet, K-sequence, Remote I/O, MODBUS master or slave
- RS422, non-isolated, distance within 1000m (3280 ft.)
- 300 / 600 / 1200 /2400 / 4800 / 9600 / 19200 / 38400 baud (DirectNet, K-sequence, MODBUS protocols),19200 / 38400 (Remote I/0 protocol)
- 8 data bits, 1 start, 1 stop bit, odd/none/even parity
- Hex / ASCII modes (use Aux function to configure)
- Selectable address 1–90 (use Aux function to configure)

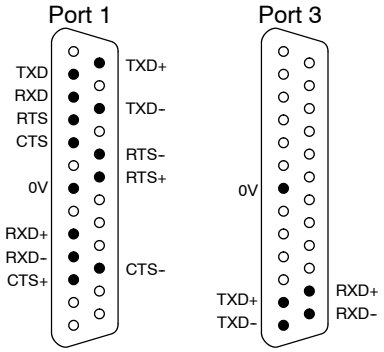


25-pin Female D Connector

Port 3 Pin Descriptions (DL450)		
1	–	Not used
2		(port 1)
3		(port 1)
4		(port 1)
5		(port 1)
6	–	Not used
7	SG	Signal ground
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)

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

The DL405 CPUs have a lithium battery to maintain 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.

Battery indicators will flash on and off when a battery needs changing.

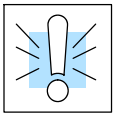
- Flashing at 2 Hz means the CPU battery needs changing.
- Fashing at 0.5 Hz means the RAM cartridge battery needs changing.

NOTE: Be sure to back up your V-memory and system parameters before replacing your CPU battery. Just save the V-memory and system parameters to either a Memory Cartridge, cassette tape, or to a personal computer (use ***Direct***SOFT).

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 proper voltage levels necessary to retain memory is fully charged.

To remove the CPU battery:

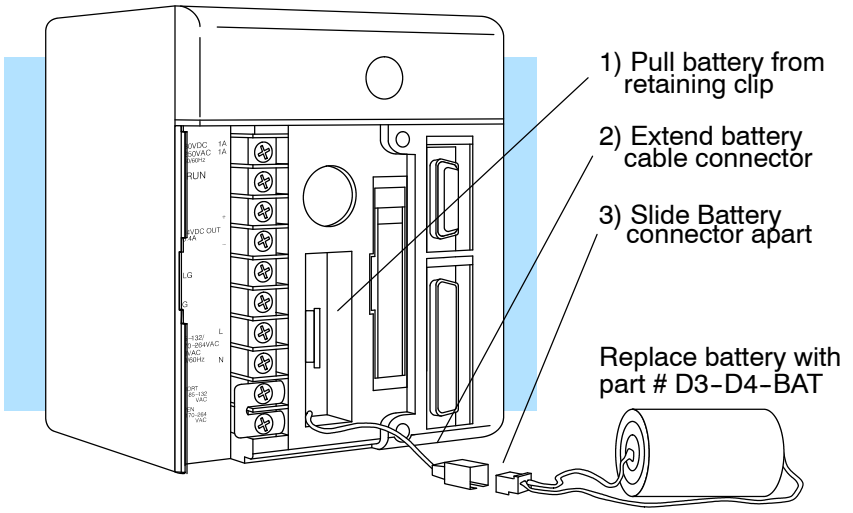
1. Pull the battery out from the battery retaining clip.
2. Lift the clip on the two wire battery connector.
3. Slide the battery connector apart.



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

To install the CPU battery:

1. Join the (keyed) battery connector so that the red wires match.
2. Push gently till the connector snaps closed
3. Slide the battery all the way into the retaining clip (flush with the opening).
4. Note the date the battery was changed.



Selecting the Program Storage Media

The DL430 CPU provides built-in EEPROM storage as a standard feature, so it cannot use other types of program storage. The DL440 CPU requires a removable memory cartridge for operation. The DL450 provides built-in FLASH memory as a standard feature. It will still accept a memory cartridge for program storage. The following paragraphs will help you choose the correct memory storage for your CPU type and application.

NOTE: The source memory (either cartridge or internal) must be specified by using either ***Direct***SOFT or the handheld programmer (D4-HPP).

Volatile and Non-volatile Memory

The two types of memory storage available are volatile and non-volatile. Volatile memory retains your data only as long as proper voltage is maintained to the storage media. Non-volatile memory does not require power to retain data. The DL405 CPUs maintain the proper voltage either through the power supply or the use of a memory backup battery.

Memory Storage Types

Memory cartridges are applicable to only the DL440 and DL450 CPUs. One cartridge is required on the DL440, but is optional on the DL450.The removable memory cartridge is available as either CMOS RAM, UVPROM or EEPROM. The RAM and EEPROM types offer a write protect jumper selection internal to the cartridge. When the cartridge is opened the jumper may be moved to the protect position to prevent accidental erasure or alterations to the program.

- CMOS RAM (Complementary Metal Oxide Semiconductor / Random Access Memory) requires a battery for memory retention, which is located inside the cartridge. The memory can be modified or changed easily with a handheld programmer or PC programming software. Battery life is typically 3 years. Refer to Chapter 9, Maintenance and Troubleshooting for battery replacement.
- UVPROM (Ultraviolet Programmable Read Only Memory) does not require a battery for memory retention, so it is classified as “non-volatile”. However, erasure (clearing memory) requires exposing the memory ICs to an ultraviolet light source. The CPU can reprogram the UVPROM cartridge after erasure, but this requires a handheld programmer.

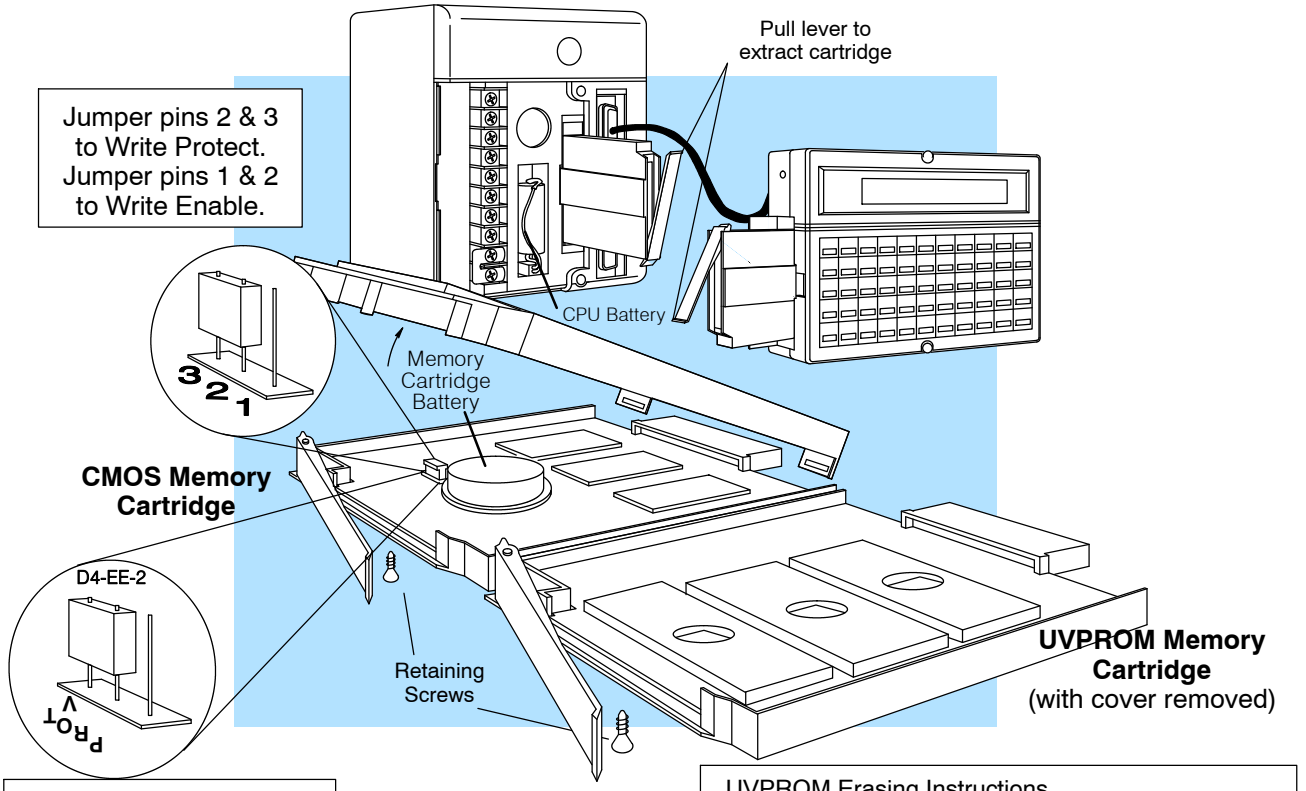
NOTE: When you purchase the UVPROM memory cartridge it is recommended you have either a RAM or an EEPROM memory cartridge for program development. Once development is completed you should then use the handheld programmer (D4-HPP) to copy your application program to the UVPROM. We recommend the UVPROM memory cartridge option for applications which are mass produced and do not require frequent alterations.

- EEPROM (Electrically Erasable Programmable Read Only Memory) does not require a battery for memory retention, so it is classified as “non-volatile”. Both erasure and programming are accomplished electrically, eliminating ultraviolet light source requirement. So, the EEPROM memory cartridge can be electrically reprogrammed (if not write protected) without being removed from the CPU. This is the memory type built into the DL430 CPU.

Memory Cartridge

The diagram below displays a Memory Cartridge for the DL440 or DL450. It shows how the memory cartridge fits in the CPU and in the handheld programmer. It also shows how to open the memory cartridge for selecting write protect (for CMOS RAM) or for erasing the UVPROM. Details about connecting the handheld programmer to the CPU is covered in DL405 Handheld Programmer Manual.

WARNING: Do not insert or remove a CPU memory cartridge while the power is connected. Your program or password may be corrupted if this occurs. A corrupted program can cause unpredictable operation which may result in a risk of injury to personnel or damage to equipment. If the password becomes corrupted, you cannot access the CPU.



Memory Cartridge Capacity Table

	D4-RAM-1	D4-RAM-2	D4-UV-2	D4-EE-2
Program Storage Capacity	7.5K	15.5K	15.5K	15.5K
Cartridge Battery Type	Lithium	Lithium	None	None
Writing Cycle Life	N/A	N/A	1000	>10,000
Write Inhibit	Internal jumper	Internal jumper	No	Internal jumper
Memory Clear Method	Electrical	Electrical	Ultraviolet light	Electrical

CPU Setup

Setting the Clock and Calendar

✗ ✓ ✓
430 440 450

The DL440 and DL450 CPUs also have a Clock / Calendar that can be used for many purposes. If you need to use this feature there are also AUX functions available that allow you set the date and time. For example, you would use AUX 52, Display/Change Calendar to set the time and date with the Handheld Programmer. With ***Direct***SOFT you would use the PLC Setup menu options.

There are also two instructions that allow you to 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 thru Saturday)
- Time — 24 hour format, Hours, Minutes, Seconds

Handheld Programmer Display

```
94/01/02 23:08:17
```

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

Variable/Fixed Scan Time Feature

✗ ✗ ✓
430 440 450

The DL450 CPU offers three type 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 10 mS to 9999 mS. The operating system inserts a delay after each ladder scan to accomplish the requested fixed scan.
- **Limit** – the PLC operates with a variable scan, but generates a watchdog timeout error if the scan time exceeds the specified amount. You can use this to trap program execution errors, for example.

To select the desired DL450 scan time option, use ***Direct***SOFT and go online with the DL450. Then select the PLC Menu, then Diagnostics, then Scantime, then Setup. The three choices of Variable, Fixed, or Limit then appear.

Password Protection

The DL405 CPUs have a password protection feature which prevents unauthorized access to CPU programs or data. Use AUX 81, 82, and 83 to either modify the password, and unlock or lock the CPU respectively. 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 (DL440/450 only) – The DL440 and DL450 feature an intermediate level of protection that you can choose by making the first character of the password the character "A". The remaining 7 characters must be numeric (0–9). The intermediate password protection differs from the standard password in that it does allow an operator interface device to access and change V-memory data such as presets. It still does not allow a ladder program edit, however.

For more information on passwords, see Appendix A, Auxiliary Functions.

Auxiliary Functions Many CPU setup tasks involve the use of Auxiliary (AUX) Functions. The AUX Functions perform many different operations, ranging from simple operating mode changes to copying programs to memory cartridges. They are divided into categories that affect different system parameters. See Appendix A for detailed descriptions of the AUX functions.

You can access the AUX Functions with the DL405 Handheld Programmer, or with ***Direct***SOFT's pull-down menus. Manuals for those products provide step-by-step instructions. Some AUX Functions are designed specifically for Handheld Programmer setup, so they are not supported by ***Direct***SOFT. The following table lists the Auxiliary functions for the different CPUs and the Handheld Programmer.

AUX Function and Description		430	440	450	HPP
AUX 1* — Operating Mode					
11	Go to Run Mode	✓	✓	✓	–
12	Go to Test Mode	✓	✓	✓	–
13	Go to Program Mode	✓	✓	✓	–
14	Run Time Edit	×	✓	✓	–
AUX 2* — RLL Operations					
21	Check Program	✓	✓	✓	–
22	Change Reference	×	✓	✓	–
23	Clear Ladder Range	✓	✓	✓	–
24	Clear Ladders	✓	✓	✓	–
25	Select MC or Flash Memory	×	×	✓	–
26	Copy MC Contents to Flash	×	×	✓	–
27	Copy Flash contents to MC	×	×	✓	–
28	Verify Flash contents = MC	×	×	✓	–
AUX 3* — V-Memory Operations					
31	Clear V Memory	✓	✓	✓	–
32	Clear V Range	✓	✓	✓	–
33	Find V-memory Value	×	✓	✓	–
AUX 4* — I/O Configuration					
41	Show I/O Configuration	✓	✓	✓	–
42	I/O Diagnostics	✓	✓	✓	–
44	Power up I/O Configuration Check	✓	✓	✓	–
45	Select Configuration	✓	✓	✓	–
46	Configure I/O	×	✓	✓	–
47	Intelligent I/O	✓	✓	✓	–

AUX Function and Description		430	440	450	HPP
AUX 5* — CPU Configuration					
51	Modify Program Name	✓	✓	✓	–
52	Display / Change Calendar	×	✓	✓	–
53	Display Scan Time	✓	✓	✓	–
54	Initialize Scratchpad	✓	✓	✓	–
55	Set Watchdog Timer	✓	✓	✓	–
56	Configure Comm. Ports	✓	✓	✓	–
57	Set Retentive Ranges	✓	✓	✓	–
58	Test Operations	✓	✓	✓	–
5C	Display Error History	×	✓	✓	–
5D	Select PLC Scan Mode	×	×	✓	–
AUX 6* — Handheld Programmer Configuration					
61	Show Revision Numbers	✓	✓	✓	
62	Beeper On / Off	×	×	×	✓
63	Backlight On / Off	×	×	×	✓
64	Select Online / Offline	×	×	×	✓
65	Run Self Diagnostics	×	×	×	✓
AUX 7* — Memory Cartridge Operations					
71	CPU to Memory Cartridge	✓	✓	✓	✓
72	Memory Cartridge to CPU	✓	✓	✓	✓
73	Compare MC with CPU	✓	✓	✓	✓
74	Memory Cart. Blank Check	×	×	×	✓
75	Clear Memory Cartridge	×	×	×	✓
76	Display Memory Cart. Type	✓	✓	✓	✓
77	Tape to Memory Cartridge	×	×	×	✓
78	Memory Cartridge to Tape	×	×	×	✓
79	Compare MC with Tape	×	×	×	✓
AUX 8* — Password Operations					
81	Modify Password	×	✓	✓	–
82	Unlock CPU	×	✓	✓	–
83	Lock CPU	×	✓	✓	–

✓ supported
× not supported
– not applicable

Clearing an Existing Program

Before entering a new program, it's a good idea to always clear ladder memory. AUX Function 24 clears the complete user program.

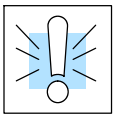
You can also use other AUX functions to clear other memory areas.

- AUX 23 — Clear Ladder Range
- AUX 31 — Clear V Memory
- AUX 32 — Clear V Range

Initializing System Memory

The DL405 CPUs maintain system parameters in a memory area often referred to as the "scratchpad". In some cases, you may make changes to the system setup that will be stored in system memory. For example, if you specify a range of Control Relays (CRs) as retentive, these changes are stored.

AUX 54 resets the system memory to the default values.



WARNING: **You may never have to use this feature unless you want to clear any 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.**
Remember, this AUX function will reset *all* system memory. If you have set special parameters such as retentive ranges, you will need to re-enter the data.

Setting the CPU Network Address

Since the DL405 CPUs have built-in ***Direct***NET ports, you can use the Handheld Programmer to set the network address for the port and the port communication parameters. The default settings are:

- Station address 1
- Hex mode
- Odd parity

The ***Direct***NET Manual provides additional information about choosing the communication settings for network operation. If you're using the bottom port for programming, you can just use the default settings. For the extra two ports on the DL450, see Chapter 4, System Design and Configuration.

Use AUX 56 to set the network address and communication parameters for the secondary port(s).

Setting Retentive Memory Ranges

The DL405 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:

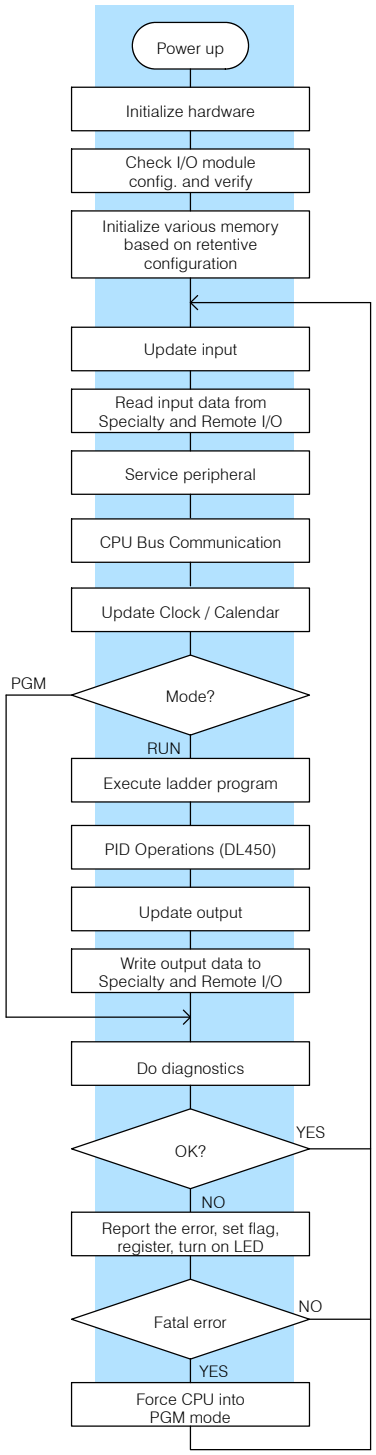
Memory Area	DL430		DL440		DL450	
	Default Range	Avail. Range	Default Range	Avail. Range	Default Range	Avail. Range
Control Relays	C600 – C737	C0 – C737	C600 – C737	C0 – C1777	C1000 – C3777	C0 – C3777
V-Memory	V2000 – V7377	V0 – V7377	V2000 – V7777	V0 – V17777	V1400 – V37777	V0 – V37777
Timers	None by default	T0 – T177	None by default	T0 – T377	None by default	T0 – T377
Counters	CT0 – CT177	CT0 – CT177	CT0 – CT177	CT0 – CT177	CT0 – CT377	CT0 – CT377
Stages	None by default	S0 – S577	None by default	S0 – S1777	None by default	S0 – S1777

Use AUX 57 to set the retentive ranges.

CPU Operation

Achieving the proper control for your equipment or process requires a good understanding of how DL405 CPUs control all aspects of system operation. The flow chart below shows the main tasks of the CPU operating system. In this section, we will investigate four aspects of CPU operation:

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



CPU Operating System

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

After the one-time powerup 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.

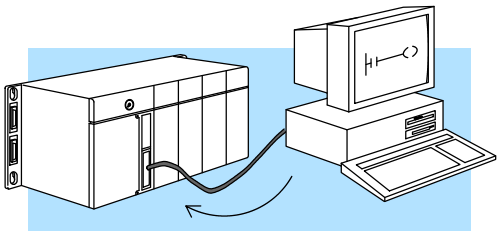
The outputs are only updated in Run mode. In program mode, they are in the off state.

In Run Mode, the CPU executes the user ladder program. Immediately afterwards, any PID loops which are configured are executed (DL450 only). 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.



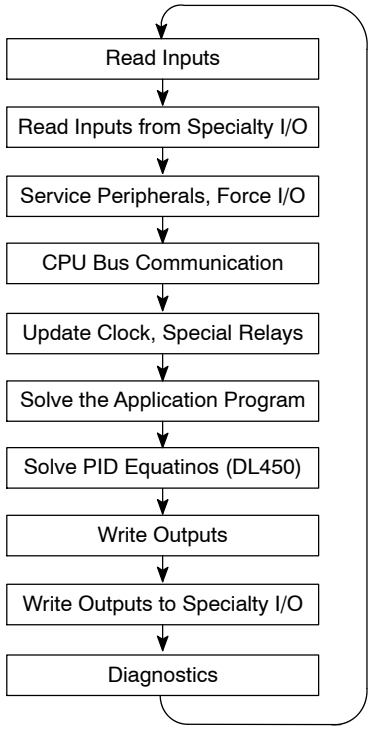
Download Program

You can use the key switch on the CPU to select Program Mode operation. Or, with the keyswitch in TERM position, you can use a programming device such as the Handheld Programmer to place the CPU in Program Mode.

Run Mode Operation

In Run Mode, the CPU executes the application program, does PID calculations for configured PID loops (DL450) only, and updates the I/O system. You can perform many operations during Run Mode. Some of these include:

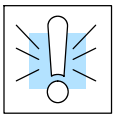
- Monitor and change I/O point status
- Update timer/counter preset values
- Update Variable memory locations



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 key switch to select Run Mode operation. Or, with the keyswitch in TERM position, you can use a programming device, such as the Handheld Programmer to place the CPU in Run Mode.

With the DL440 or DL450, you can also edit the program during Run Mode. The Run Mode Edits are not “bumpless.” Instead, the CPU maintains the outputs in their last state while it accepts the new program information. If an error is found in the new program, then the CPU will turn all the outputs off and enter the Program Mode.



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

Read Inputs

The CPU reads the status of all inputs, then stores it in the image register. Input image register locations are designated with an X followed by a memory location. Image register data is used by the CPU when it solves the application program.

Of course, an input may change *after* the CPU has just read the inputs. Generally, the CPU scan time is measured in milliseconds. If you have an application that cannot wait until the next I/O update, you can use Immediate Instructions. These do not use the status of the input image register to solve the application program. The Immediate instructions immediately read the input status directly from I/O modules. However, this lengthens the program scan since the CPU has to read the I/O point status again. A complete list of the Immediate instructions is included in Chapter 5.

Read Inputs from Specialty and Remote I/O

After the CPU reads the inputs from the input modules, it reads any input point data from any Specialty modules that are installed, such as High Speed Counter modules, etc.This is also the portion of the scan that reads the input status from Remote I/O racks. The GX data type is used for both remote inputs and outputs. (The DL405 Remote I/O Manual provides additional information on how to set up the remote I/O link.)

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

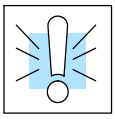
After the CPU reads the inputs from the input modules, it reads any attached peripheral devices. This is primarily a communications service for any attached devices. For example, it would read a programming device to see if any input, output, or other memory type status needs to be modified.

Forced I/O— temporarily changes 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 just need to force a bit on to trigger another event.

Forced Inputs — The CPU reads the status of X inputs during the Read Inputs portion of the scan. When the CPU services the programming device, it logs any request to force an X input on. If the input is used in the application program, the ladder X contact is considered closed (on). Since an X input is a real-world input point, the CPU will change the status when it reads the inputs on the next scan.

Forced Outputs— Outputs which are not used in the program can be forced on and off for troubleshooting and maintenance purposes. You can temporarily allow the forcing of any output by inserting an END coil instruction at the beginning of the ladder program. Then you can use ***Direct***SOFT or a HPP to force outputs on and off.

The DL405 CPUs only retain the forced value for one scan, if the I/O point corresponds to an actual point on a module in the system. However, if the point address is greater than any actual I/O point address in the system or it is not used in the ladder program, then the point will maintain the forced status.



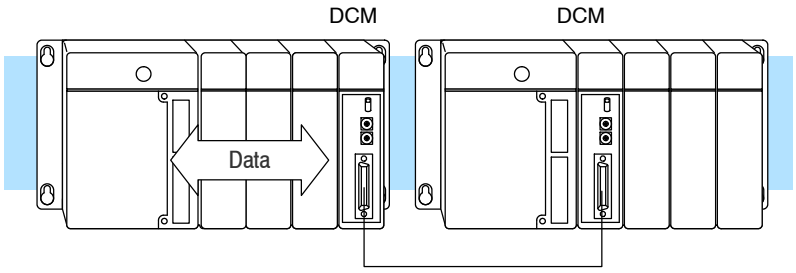
WARNING: Only authorized personnel fully familiar with all aspects of the application should make changes to the program. Make sure you thoroughly consider the impact of any changes to minimize the risk of personal injury or damage to equipment.

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 CoProcessor 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.



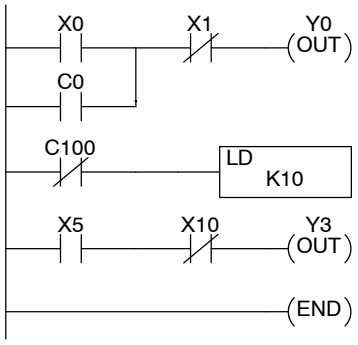
Update Clock, Special Relays, and Special Registers

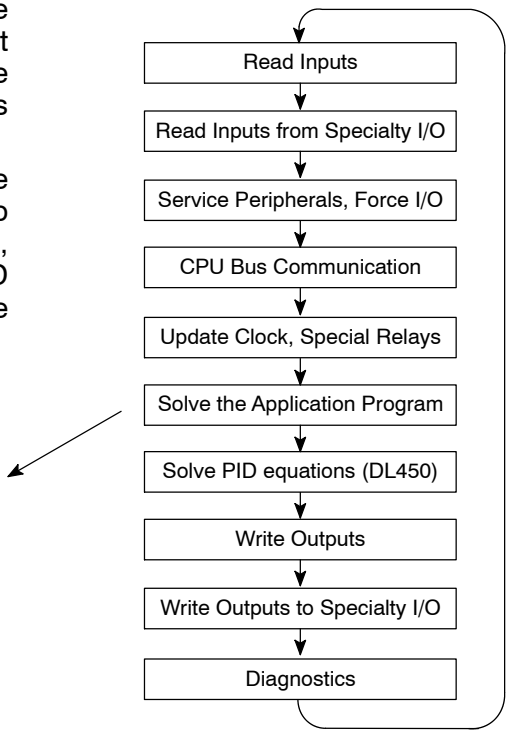
The DL440 and DL450 CPUs have an internal real-time clock and calendar timer which is accessible to the application program. Special V-memory locations hold this information. This portion of the execution cycle makes sure these locations get updated on every scan. Also, there are several different Special 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 system outputs.

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





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

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

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

Solve PID Loop Equations

✗	✗	✓
430	440	450

The DL450 CPU 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 update the actual remote modules during the next communication sequence between the master and slave modules. Remember, the Remote I/O link communication is managed by the Remote Master, not the CPU.

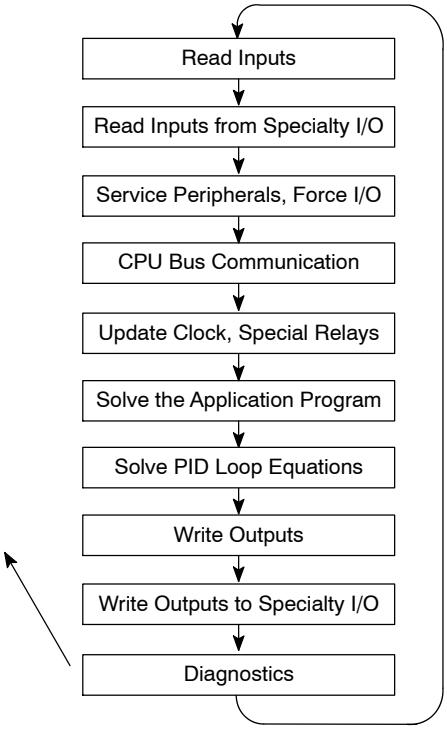
Diagnostics

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

- calculating the scan time
- updating special relays
- resetting the watchdog timer

DL405 CPUs automatically detect and report many different error conditions. Appendix B 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. DL405 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 200 mS. If this time is exceeded the CPU will enter the Program Mode, turn off all outputs, and report the error. For example, the Handheld Programmer displays “E003 S/W TIMEOUT” when the scan overrun occurs.



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

I/O Response Time

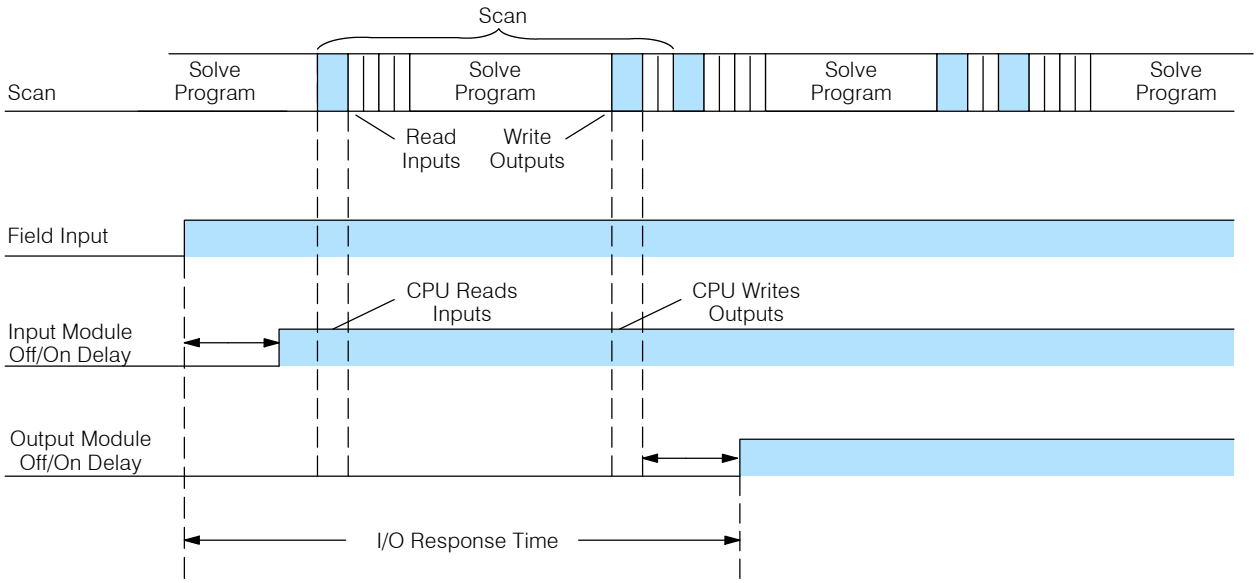
Is Timing Important for Your Application?

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



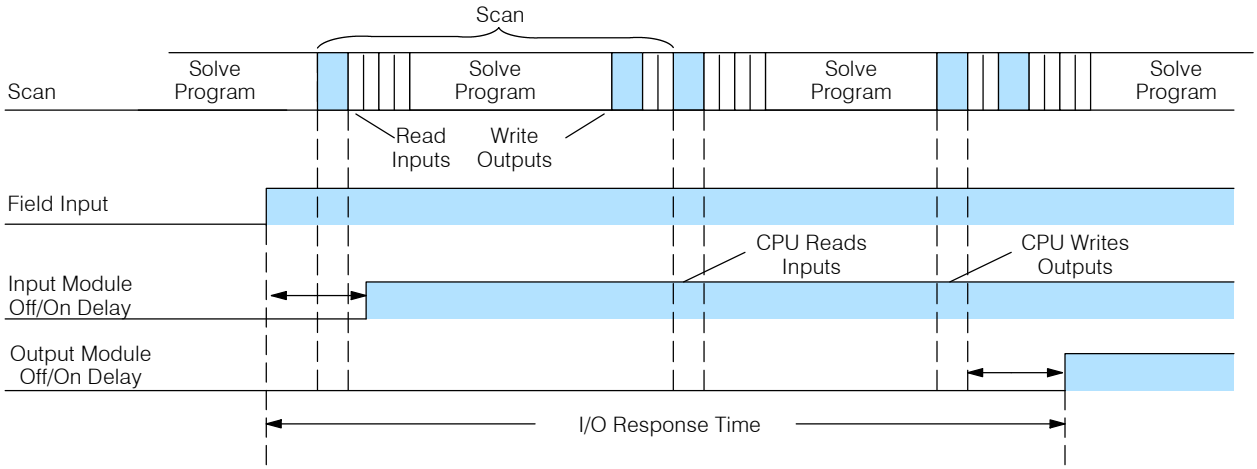
In this case, you can calculate the response time by simply adding the following items.

Input Delay + Scan Time + Output Delay = Response Time

Normal Maximum I/O Response

The I/O response time is longest when the module 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

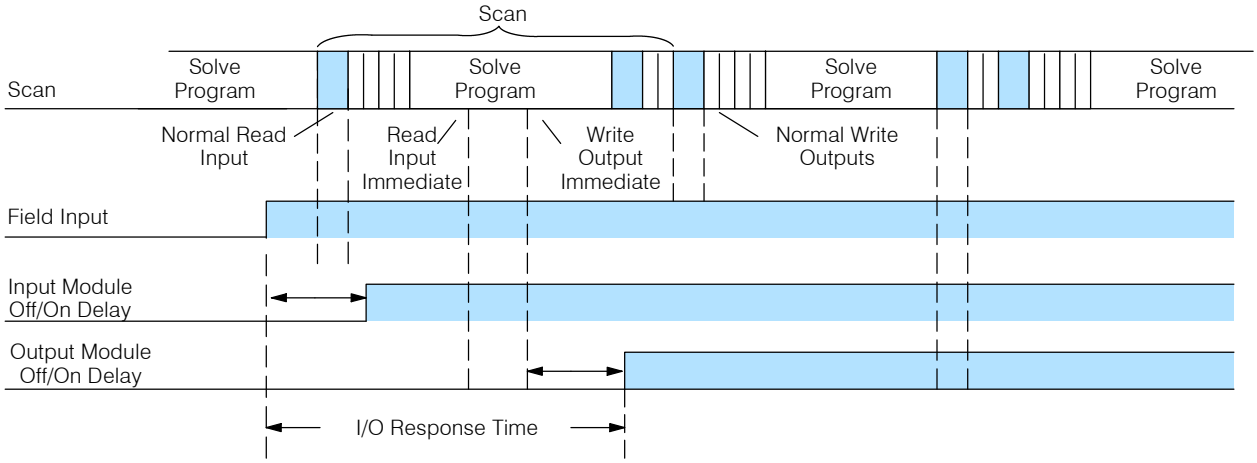


Improving Response Time

There are a few things you can do the 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 inbetween.

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

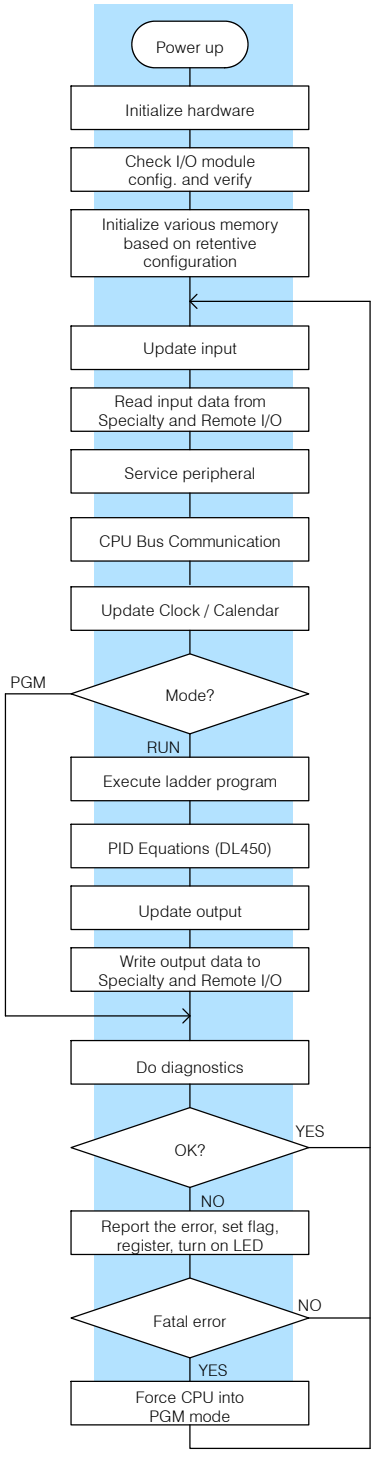
The scan time covers all the cyclical tasks that are performed by the operating system. You can use ***Direct***SOFT or the Handheld Programmer to display the minimum, maximum, and current scan times that have occurred since the previous Program Mode to Run Mode transition. This information can be very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

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

Your choice of I/O modules and system configuration, such as expansion or remote I/O, can also affect the scan time. However, 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 signal wires for each individual I/O point.

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



Initialization Process

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

Initialization	DL430	DL440	DL450
Minimum Time	1.2 Seconds	1.0 Seconds	1.9 Seconds
Maximum Time	3.2 Seconds	2.5 Seconds	3.3 Seconds

Reading Inputs

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

Timing Factors	DL430	DL440	DL450
Overhead	20.0 µs	14.5 µs	20.0 µs
Per input module	48.0 µs	22.6 µs	13.0 µs
Per input point	4.0 µs	2.5 µs	6.3 µs

For example, the time required for a DL430 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 = 20µs + (48µs x NM) + (4µs x NI)

Example

Time = 20µs + (48µs x 2) + (4µs x 16)

Time = 180 µs

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

Reading Inputs from Specialty I/O

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

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

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

Specialty Module	DL430	DL440	DL450
Overhead	32.0 µs	20.0 µs	20.0 µs
Per module (with inputs)	100.0 µs	67.0 µs	13.0 µs
Per input point	80.0 µs	54.0 µs	13.8 µs
Remote Module	**DL430**	**DL440**	**DL450**
Overhead	32.0 µs	22.0 µs	19.0 µs
Per module (with inputs)	150.0 µs	100.0 µs	62.0 µs
Per input point	25.0 µs	17.0 µs	11.2 µs

For example, the time required for a DL430 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 total number of input points.)

Remote I/O	High-Speed Counter
Formula	**Formula**
Time = 32µs + (150µs x NM) + (25µs x NI)	Time = 32µs + (100µs x NM) + (80µs x NI)
Example	**Example**
Time = 32µs + (150µs x 2) + (25µs x 32)	Time = 32µs + (100µs x 1) + (80µs x 16)
Time = 1832 µs	Time = 1412 µs

Total Time Time = 3244 µs

Service Peripherals

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

To Log Request (anytime)		DL430	DL440	DL450
Nothing Connected	Min. & Max.	0 µs	0 µs	0 µs
Port 0	Send Min. / Max.	52 / 62 µs	40 / 48 µs	38 / 38 µs
	Rec. Min. / Max.	60 / 78 µs	52 / 63 µs	45 /45 µs
Port 1	Send Min. / Max.	60 / 78 µs	46 / 50 µs	41 / 48 µs
	Rec. Min. / Max.	68 / 86 µs	66 / 70 µs	47 / 59 µs
Port 2	Send Min. / Max.	N/A	N/A	41 / 48 µs
	Rec. Min. / Max.	N/A	N/A	47 / 59 µs
Port 3	Send Min. / Max.	N/A	N/A	38 / 38 µs
	Rec. Min. / Max.	N/A	N/A	45 / 45 µs

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

To Service Request	DL430	DL440	DL450
Minimum	170 µs	120 µs	96 µs
Run Mode Max.	18 ms	26 ms	160 ms
Program Mode Max.	3 Seconds	15 Seconds	11.2 Seconds

CPU Bus Communication

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

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

Update Clock/Calendar, Special Relays, Special Registers

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

Modes		DL430	DL440	DL450
Program Mode	Minimum	8.0 µs fixed	35.0 µs	12.0 µs
	Maximum	8.0 µs fixed	48.0 µs	12.0 µs
Run Mode	Minimum	20.0 µs	60.0 µs	22.0 µs
	Maximum	26.0 µs	85.0 µs	29.0 µs

Writing Outputs

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

Timing Factors	DL430	DL440	DL450
Overhead	20.0 µs	12.6 µs	15.0 µs
Per output module	45.0 µs	21.0 µs	13.0 µs
Per output point	4.0 µs	2.5 µs	14.1 µs

For example, the time required for a DL430 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 total number of output points).

Formula

Time = 20µs + (45µs x NM) + (4µs x NO)

Example

Time = 20µs + (45µs x 2) + (4µs x 32)

Time = 238 µs

Writing Outputs to Specialty I/O

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

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

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

Specialty Module	DL430	DL440	DL450
Overhead	32.0 µs	20.0 µs	18.0 µs
Per module (with outputs)	100.0 µs	67.0 µs	13.0 µs
Per output point	80.0 µs	54.0 µs	14.1 µs
Remote Module	**DL430**	**DL440**	**DL450**
Overhead	32.0 µs	22.0 µs	15.0 µs
Per module (with outputs)	150.0 µs	100.0 µs	54.0 µs
Per output point	25.0 µs	17.0 µs	13.9 µs

For example, the time required for a DL430 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 NO is the total number of output points.)

Remote I/O

Formula
Time = 32µs + (150µs x NM) + (25µs x NO)

Example
Time = 32µs + (150µs x 2) + (25µs x 32)

Time = 1832 µs

High-Speed Counter

Formula
Time = 32µs + (100µs x NM) + (80µs x NO)

Example
Time = 32µs + (100µs x 1) + (80µs x 4)

Time = 452 µs

Total Time

Time = 2284 µs

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

Diagnostics

The DL405 CPUs perform many types of system diagnostics. The amount of time required depends on many things, such as the number of I/O modules installed, etc. The following table shows the minimum and maximum times that can be expected.

Diagnostic Time	DL430	DL440	DL450
Minimum	680.0 µs	540.0 µs	282.0 µs
Maximum	880.0 µs	920.0 µs	398.0 µs

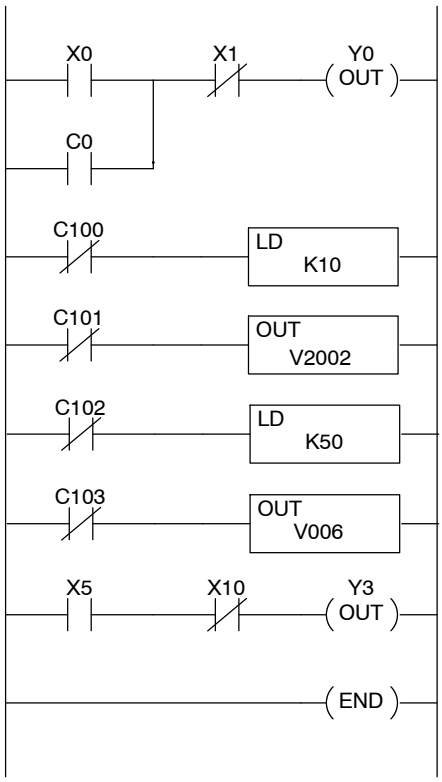
Application Program Execution

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 DL430 running the program shown would be calculated as follows.



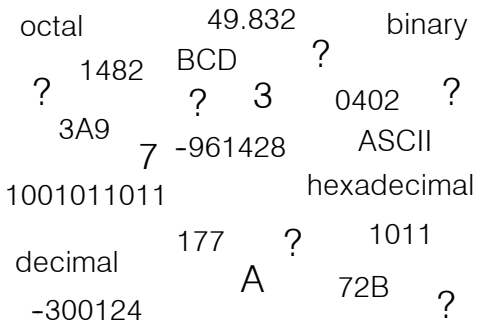
Instruction		Time
STR X0		4.7µs
OR C0		3.1µs
ANDN X1		3.4µs
OUT Y0		8.3µs
STRN C100	5.7µs	
LD K10		112.0µs
STRN C101	8.3µs	
OUT V2002	26.0µs	
STRN C102	8.3µs	
LD K50		112.0µs
STRN C103	8.3µs	
OUT V0006	26.0µs	
STR X5		4.7µs
ANDN X10		3.4µs
OUT Y3		8.3µs
END		15.5µs
TOTAL		358µs

Appendix C provides a complete list of instruction execution times for DL405 CPUs.

Program Control Instructions — the DL440 and DL450 CPUs offer additional instructions that can change the way the program executes. These instructions include FOR/NEXT loops, Subroutines, and Interrupt Routines. These instructions can interrupt the normal program flow and effect 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 **AutomationDirect** PLCs. The information you learn here applies to all our PLCs.

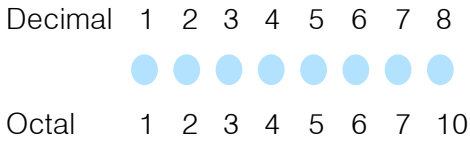


As any good computer does, PLCs store and manipulate numbers in binary form: just ones and zeros. So why do we have to deal with numbers in so many different forms? Numbers have meaning, and some *representations* are more convenient than others for particular purposes. Sometimes we use numbers to represent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

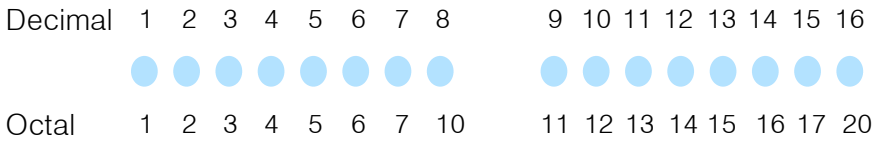
PLC Resources

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

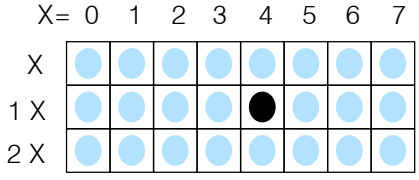


In the figure below, we have two groups of eight circles. Counting in octal we have "20" items, meaning 2 groups of eight, plus 0 individuals Don't say "twenty", say "two-zero octal". This makes a clear distinction between number systems.



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

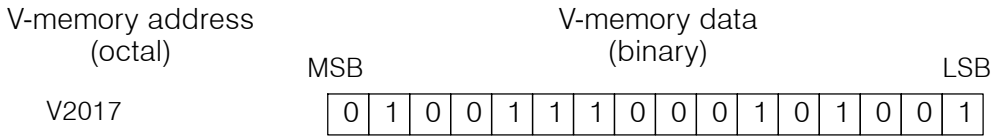
Our circles are in an array of square containers to the right. To access a resource, our PLC instruction will address its location using the octal references shown. If these were counters, "CT14" would access the black circle location.



V–Memory

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

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

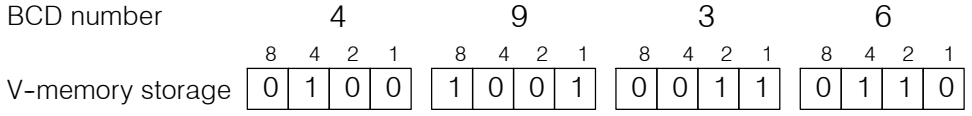


V-memory data is 16-bit binary, but we rarely program the data registers one bit at a time. We use instructions or viewing tools that let us work with binary, decimal, octal, and hexadecimal numbers. All these are converted and stored as binary for us.

A frequently-asked question is "How do I tell if a number is binary, octal, BCD, or hex"? The answer is that we usually cannot tell 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.



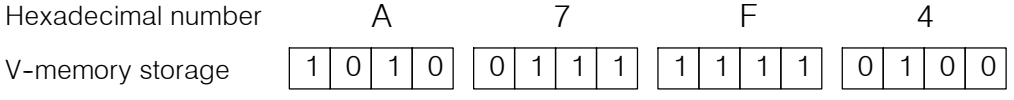
In a pure binary sense, a 16-bit word represents numbers from 0 to 65535. In storing BCD numbers, the range is reduced to 0 to 9999. Many math instructions use BCD data, and ***Direct***SOFT and the handheld programmer allow us to enter and view data in BCD. Special RLL instructions convert from BCD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

Hexadecimal Numbers

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

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

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



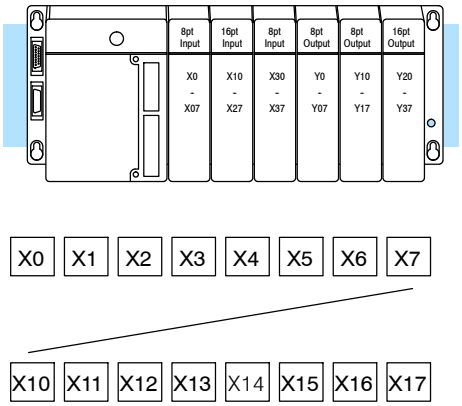
Memory Map

With any PLC system, you generally have many different types of information to process. This includes input device status, output device status, various timing elements, parts counts, etc. It is important to understand how the system represents and stores the various types of data. For example, you need to know how the system identifies input points, output points, data words, etc. The following paragraphs discuss the various memory types used in the DL405 CPUs. Memory maps for all DL405 CPUs follow the memory descriptions.

Octal Numbering System

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.

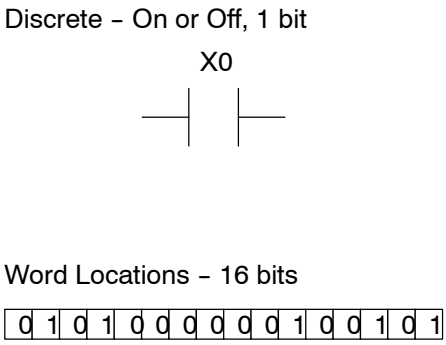
Refer to the previous section on PLC Numbering Systems for more on octal numbering.



Discrete and Word Locations

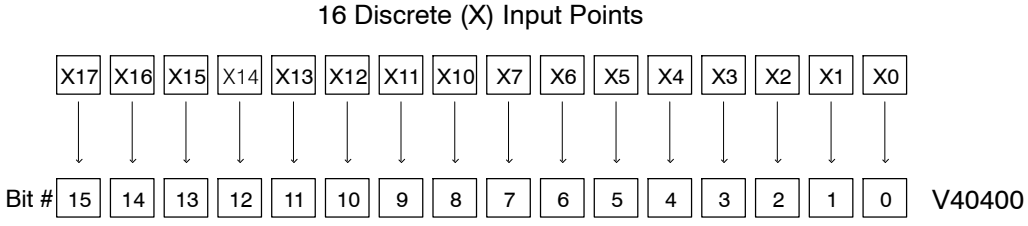
As you examine the different memory types, you'll notice two types of memory in the DL405, discrete and word memory. Discrete memory is one bit that can be either a 1 or a 0. Word memory is referred to as V-memory (variable) and is a 16-bit location normally used to manipulate data/numbers, store data/numbers, etc.

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



V-Memory Locations for Discrete Memory Areas

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



These discrete memory areas and the corresponding V-memory locations are listed in the Memory Map tables for the DL430, DL440, and DL450 CPUs in this chapter.

Input Points (X Data Type)

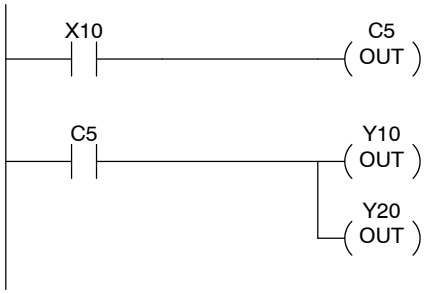
The discrete input points are noted by an X data type. Refer to the memory maps for the number of discrete input points for your CPU type. In this example, the output point Y0 will energize when input X0 turns on.

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 type. In this example, output point Y1 will energize when input X1 turns on.

Control Relays (C Data Type)

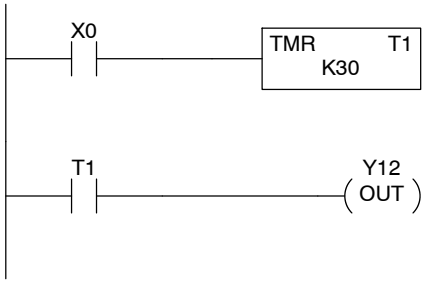
Control relays are discrete bits normally used to control the user program. The control relays do not represent a real world device, that is, they cannot be physically tied to switches, output coils, etc. They are internal to the CPU. Because of this, control relays can be programmed as discrete inputs 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 X10 turns on. The second rung shows a simple example of how to use a control relay as an input.

Timers and Timer Status Bits (T Data type)

The amount of timers available depends on the model of CPU you are using. The tables at the end of this section provide the amount of timers in each DL405 CPU 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 bit will be on when the current value is equal or greater than the preset value of a corresponding timer.



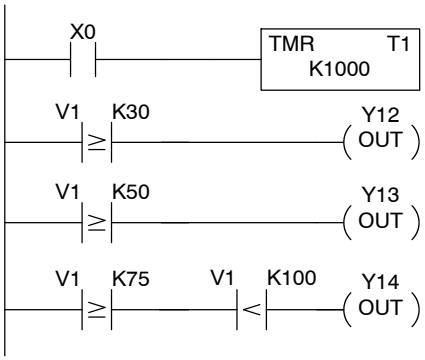
When input X0 turns on, timer T1 will start. When the timer reaches the preset of 3 seconds (K of 30) timer status contact T1 turns on. When T1 turns on, output Y12 energizes.

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)

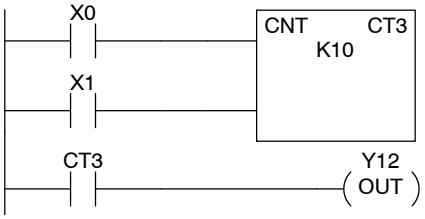
Some information is automatically stored in V-memory, such as the current values associated with timers. For example, V0 holds the current value for Timer 0, V1 holds the current value for Timer 1, etc. These are 4-digit BCD values.

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



Counters and Counter Status Bits (CT Data type)

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

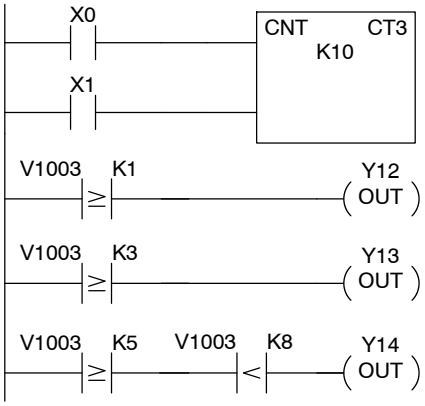


Each time contact X0 transitions from off to on, the counter increments by one. (If X1 comes on, the counter is reset to zero.) When the counter reaches the preset of 10 counts (K of 10) counter status contact CT3 turns on. When CT3 turns on, output Y12 turns on.

Counter Current Values (V Data Type)

Just like the timers, the counter current values are also automatically stored in V-memory. For example, V1000 holds the current value for Counter CT0, V1001 holds the current value for Counter CT1, etc. These are 4-digit BCD values.

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

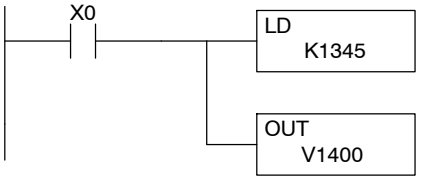


Word Memory (V Data Type)

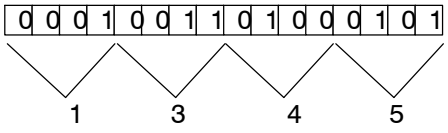
Word memory is referred to as V-memory (variable) and is a 16-bit location normally used to manipulate data/numbers, store data/numbers, etc.

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

The example shows how a four-digit BCD constant is loaded into the accumulator and then stored in a V-memory location.



Word Locations – 16 bits

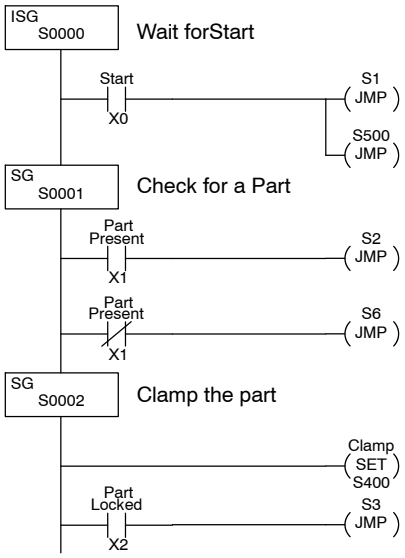


Stages (S Data type)

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

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

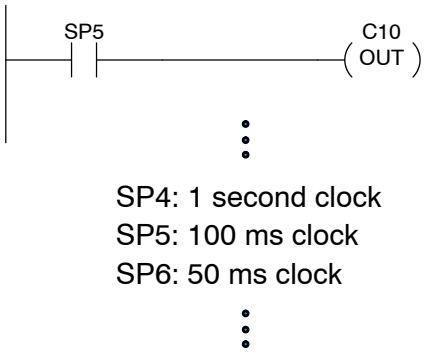
Ladder Representation



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. Appendix D provides a complete listing of the special relays.

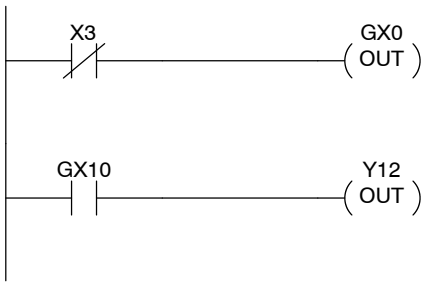
In this example, control relay C10 will energize for 50 ms and de-energize for 50 ms because SP5 is a pre-defined relay that will be on for 50 ms and off for 50 ms.



Remote I/O Points (GX Data Type)

Remote I/O points are represented by global relays. They are generally used only to control remote I/O, but they can be used as normal control relays when remote I/O is not used in the system. You should notice the same data type, GX, is used for both the remote input and remote output points. 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 GX0 represents an output point and memory location GX10 represents an input point.



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
V700, V701	Contains a copy of the contents of the accumulator. V700 is the lower 16 bit word, V701 is the upper 16 bit word.
V702, V703	Contains a copy of the contents of the 1st data stack location. V702 is the lower 16 bit word, V703 is the upper 16 bit word.
V704, V705	Contains a copy of the contents of the 2nd data stack location. V704 is the lower 16 bit word, V705 is the upper 16 bit word.
V706, V707	Contains a copy of the contents of the 3rd data stack location. V705 is the lower 16 bit word, V706 is the upper 16 bit word.
V710, V711	Contains a copy of the contents of the 4th data stack location. V710 is the lower 16 bit word, V711 is the upper 16 bit word.
V712, V713	Contains a copy of the contents of the 5th data stack location. V712 is the lower 16 bit word, V713 is the upper 16 bit word.
V714, V715	Contains a copy of the contents of the 6th data stack location. V714 is the lower 16 bit word, V715 is the upper 16 bit word.
V716, V717	Contains a copy of the contents of the 7th data stack location. V716 is the lower 16 bit word, V717 is the upper 16 bit word.
V720, V721	Contains a copy of the contents of the 8th data stack location. V720 is the lower 16 bit word, V721 is the upper 16 bit word.

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	450: Battery voltage in tenths of a volt, (e.g., V7746 = 0031 is 3.1 Volts).

✓ ✓ ✓
430 440 450

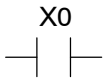
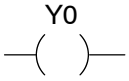
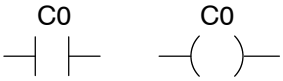
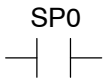
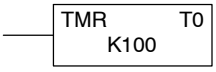
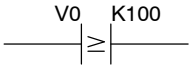
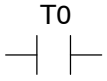
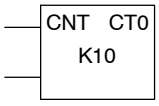
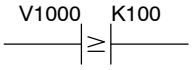
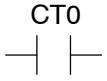
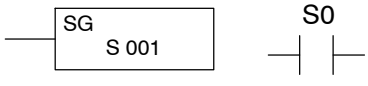
System V-memory (continued)	Description of Contents
V7751	Fault Message Error Code — stores the 4-digit BCD code used with the FAULT instruction when the instruction is executed. If you've used ASCII messages (DL440/DL450 only) then the data label (DLBL) reference number for that message is stored here.
V7752	I/O configuration Error — stores the module ID code for the module that does not match the current configuration.
V7753	I/O Configuration Error — stores the correct module ID code.
V7754	I/O Configuration Error — identifies the base and slot number.
V7755	Error code — stores the fatal error code.
V7756	Error code — stores the major error code.
V7757	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.

The following system control relays are valid only for DL450 CPU remote I/O setup on Communications Port 3.

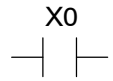
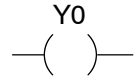
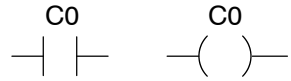
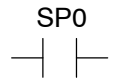
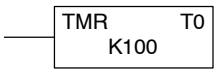
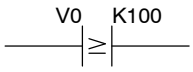
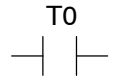
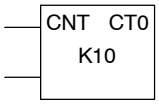
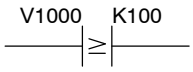
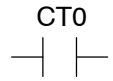
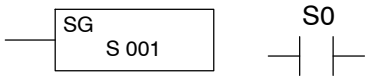
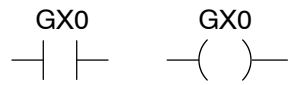
× × ✓
430 440 450

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

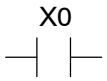
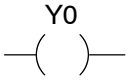
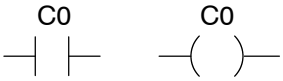
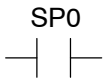
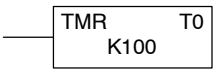
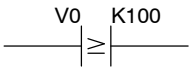
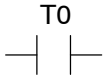
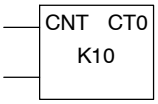
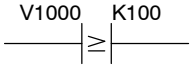
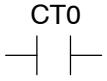
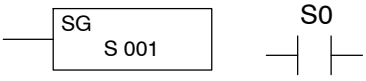
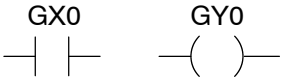
DL430 Memory Map

Memory Type	Discrete Memory Reference (octal)	Word Memory Reference (octal)	Qty. Decimal	Symbol
Input Points	X0 – X477	V40400 – V40423	320	
Output Points	Y0 – Y477	V40500 – V40523	320	
Control Relays	C0 – C737	V40600 – V40635	512	
Special Relays	SP0 – SP137 SP320 – SP617	V41200 – V41205 V41215 – V41230	288	
Timers	T0 – T177	None	128	
Timer Current Values	None	V00000 – V00177	128	
Timer Status Bits	T0 – T177	V41100 – V41107	128	
Counters	CT0 – CT177	None	128	
Counter Current Values	None	V01000 – V01177	128	
Counter Status Bits	CT0 – CT177	V41140 – V41147	128	
User Data Words	None	V1400 – V7377	3072	None specific, used with many instructions
Stages	S0 – S577	V41000 – V41027	384	
Remote In / Out	GX0 – GX777	V40000 – V40037	512	
System parameters	None	V7400 – V7777	256	None specific, used with many instructions

DL440 Memory Map

Memory Type	Discrete Memory Reference (octal)	Word Memory Reference (octal)	Qty. Decimal	Symbol
Input Points	X0 – X477	V40400 – V40423	320	
Output Points	Y0 – Y477	V40500–V40523	320	
Control Relays	C0 – C1777	V40600–V40677	1024	
Special Relays	SP0 – SP137 SP320 – SP717	V41200–V41205 V41215–V41234	352	
Timers	T0 – T377	None	256	
Timer Current Values	None	V00000 – V00377	256	
Timer Status Bits	T0 – T377	V41100 – V41117	256	
Counters	CT0 – CT177	None	128	
Counter Current Values	None	V01000 – V01177	128	
Counter Status Bits	CT0 – CT177	V41140 – V41147	128	
User Data Words	None	V1400 – V7377 V10000 – V17777	3072 4096	None specific, used with many instructions
Stages	S0 – S1777	V41000 – V41077	1024	
Remote In / Out	GX0 – GX1777	V40000 – V40077	1024	
System parameters	None	V700 – V737 V7400 – V7777	288	None specific, used with many instructions

DL450 Memory Map

Memory Type	Discrete Memory Reference (octal)	Word Memory Reference (octal)	Qty. Decimal	Symbol
Input Points	X0 – X1777	V40400 – V40477	1024	
Output Points	Y0 – Y1777	V40500 – V40577	1024	
Control Relays	C0 – C3777	V40600 – V40777	2048	
Special Relays	SP0 – SP777	V41200 – V41237	512	
Timers	T0 – T377	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 Words	None	V1400 – V7377 V10000 – V36777	3072 11776	None specific, used 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, used 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 the alias is optional, but some users may find the alias to be helpful when developing a program. The table below shows how the aliases can be used to reference memory locations.

NOTE: Ranges depend on CPU type.

Address Start	Alias Start	Example
V0	TA0	V0 is the timer accumulator value for timer 0, therefore, it's 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, it's alias is VGX20.
V40200 (DL450 only)	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, it's 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, it's 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, it's 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, it's 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, it's 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, it's 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, it's alias is VSP20.

X Input/Y Output Bit Map

This table provides a listing of individual Input and Output points associated with each V-memory address bit for the DL430, DL440, and DL450 CPUs.

MSB DL430/DL440/DL450 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

MSB DL450 Additional 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		
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								DL450 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					DL430 / DL440 / DL450 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

This portion of the table shows additional Control Relays points available with the DL440 and DL450.

MSB					DL440 / DL450 Additional Control Relays (C)											LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740		V40636
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760		V40637
1017	1016	1015	1014	1013	1012	1011	1010	1007	1006	1005	1004	1003	1002	1001	1000		V40640
1037	1036	1035	1034	1033	1032	1031	1030	1027	1026	1025	1024	1023	1022	1021	1020		V40641
1057	1056	1055	1054	1053	1052	1051	1050	1047	1046	1045	1044	1043	1042	1041	1040		V40642
1077	1076	1075	1074	1073	1072	1071	1070	1067	1066	1065	1064	1063	1062	1061	1060		V40643
1117	1116	1115	1114	1113	1112	1111	1110	1107	1106	1105	1104	1103	1102	1101	1100		V40644
1137	1136	1135	1134	1133	1132	1131	1130	1127	1126	1125	1124	1123	1122	1121	1120		V40645
1157	1156	1155	1154	1153	1152	1151	1150	1147	1146	1145	1144	1143	1142	1141	1140		V40646
1177	1176	1175	1174	1173	1172	1171	1170	1167	1166	1165	1164	1163	1162	1161	1160		V40647
1217	1216	1215	1214	1213	1212	1211	1210	1207	1206	1205	1204	1203	1202	1201	1200		V40650
1237	1236	1235	1234	1233	1232	1231	1230	1227	1226	1225	1224	1223	1222	1221	1220		V40651
1257	1256	1255	1254	1253	1252	1251	1250	1247	1246	1245	1244	1243	1242	1241	1240		V40652
1277	1276	1275	1274	1273	1272	1271	1270	1267	1266	1265	1264	1263	1262	1261	1260		V40653
1317	1316	1315	1314	1313	1312	1311	1310	1307	1306	1305	1304	1303	1302	1301	1300		V40654
1337	1336	1335	1334	1333	1332	1331	1330	1327	1326	1325	1324	1323	1322	1321	1320		V40655
1357	1356	1355	1354	1353	1352	1351	1350	1347	1346	1345	1344	1343	1342	1341	1340		V40656
1377	1376	1375	1374	1373	1372	1371	1370	1367	1366	1365	1364	1363	1362	1361	1360		V40657
1417	1416	1415	1414	1413	1412	1411	1410	1407	1406	1405	1404	1403	1402	1401	1400		V40660
1437	1436	1435	1434	1433	1432	1431	1430	1427	1426	1425	1424	1423	1422	1421	1420		V40661
1457	1456	1455	1454	1453	1452	1451	1450	1447	1446	1445	1444	1443	1442	1441	1440		V40662
1477	1476	1475	1474	1473	1472	1471	1470	1467	1466	1465	1464	1463	1462	1461	1460		V40663
1517	1516	1515	1514	1513	1512	1511	1510	1507	1506	1505	1504	1503	1502	1501	1500		V40664
1537	1536	1535	1534	1533	1532	1531	1530	1527	1526	1525	1524	1523	1522	1521	1520		V40665
1557	1556	1555	1554	1553	1552	1551	1550	1547	1546	1545	1544	1543	1542	1541	1540		V40666
1577	1576	1575	1574	1573	1572	1571	1570	1567	1566	1565	1564	1563	1562	1561	1560		V40667
1617	1616	1615	1614	1613	1612	1611	1610	1607	1606	1605	1604	1603	1602	1601	1600		V40670
1637	1636	1635	1634	1633	1632	1631	1630	1627	1626	1625	1624	1623	1622	1621	1620		V40671
1657	1656	1655	1654	1653	1652	1651	1650	1647	1646	1645	1644	1643	1642	1641	1640		V40672
1677	1676	1675	1674	1673	1672	1671	1670	1667	1666	1665	1664	1663	1662	1661	1660		V40673
1717	1716	1715	1714	1713	1712	1711	1710	1707	1706	1705	1704	1703	1702	1701	1700		V40674
1737	1736	1735	1734	1733	1732	1731	1730	1727	1726	1725	1724	1723	1722	1721	1720		V40675
1757	1756	1755	1754	1753	1752	1751	1750	1747	1746	1745	1744	1743	1742	1741	1740		V40676
1777	1776	1775	1774	1773	1772	1771	1770	1767	1766	1765	1764	1763	1762	1761	1760		V40677

This portion of the table shows additional Control Relays points available with the DL450.

MSB							DL450 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								DL450 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 Maps

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

MSB DL430 / DL440 / DL450 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 DL440 and DL450.

MSB DL440 / DL450 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 DL450.

MSB DL450 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 the individual remote I/O points associated with each V-memory address bit. The DL430 and DL440 CPUs use the GX point type for both remote input and output point types. The DL450 CPU has the additional GY point type for use as remote output point references.

MSB DL430 / DL440 / DL450 Remote I/O (GX) and (GY) Point														LSB		GX Address	GY Address (DL450)
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 available with the DL440 and DL450. The (GY) remote output point type is available only with the DL450 (the GX type points works as both input and output point types for the DL440).

MSB								DL440 / DL450 Additional Remote I/O (GX) Points							LSB	GX Address	GY Address (DL450)
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

This portion of the table shows additional Remote I/O (GX and GY) points available with the DL450.

MSB					DL450 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

This table is continued from the previous page.

MSB					DL450 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 the individual stage control bits associated with each V-memory address bit.

MSB					DL430 / DL440 / DL450 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

MSB					DL440 / DL450 Additional Stage (S) Control Bits											LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
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								DL440 / DL450 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

4

System Design and Configuration

In This Chapter. . . .

- DL405 System Design Strategies
- Module Placement and Configuration
- Calculating the Power Budget
- Local I/O Expansion
- Remote I/O Expansion
- Network Connections to MODBUS® and *Direct*NET
- Network Slave Operation
- Network Master Operation

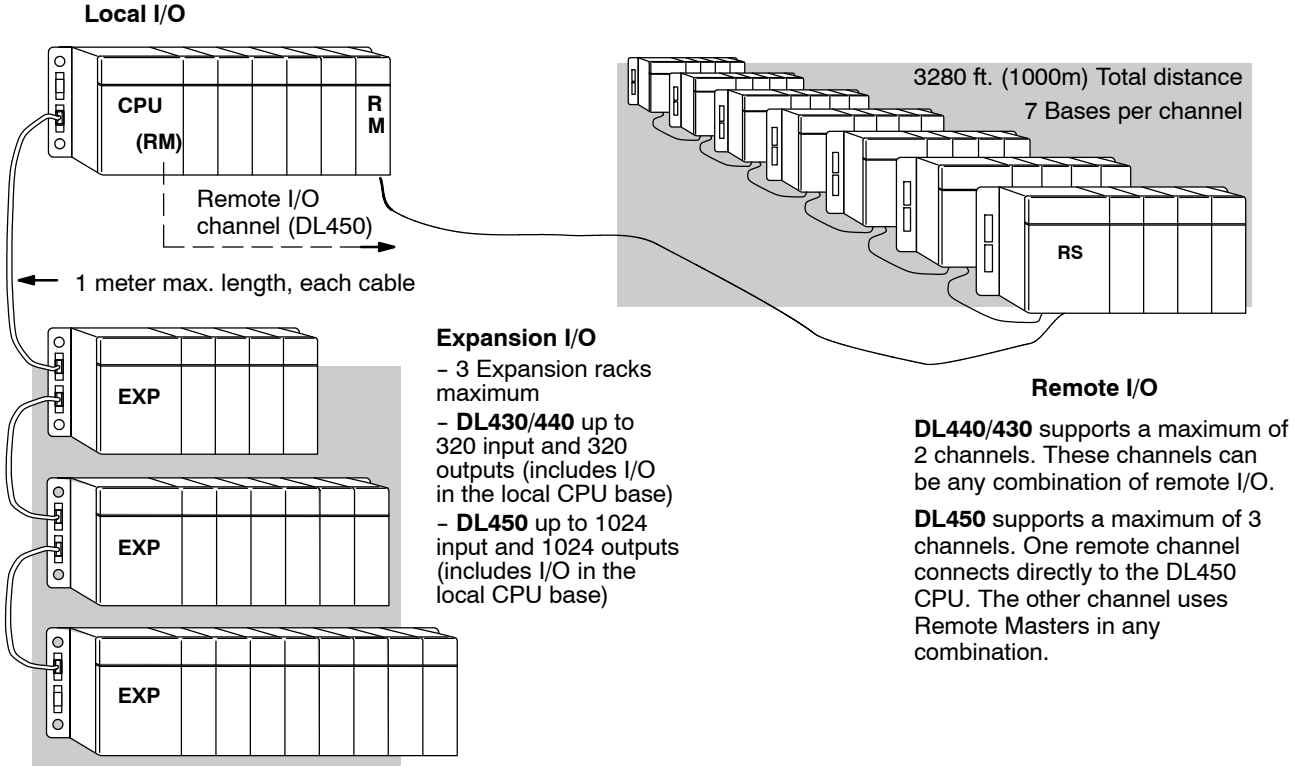
DL405 System Design Strategies

I/O System Configurations

The DL405 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 to the 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 a DL450 CPU.

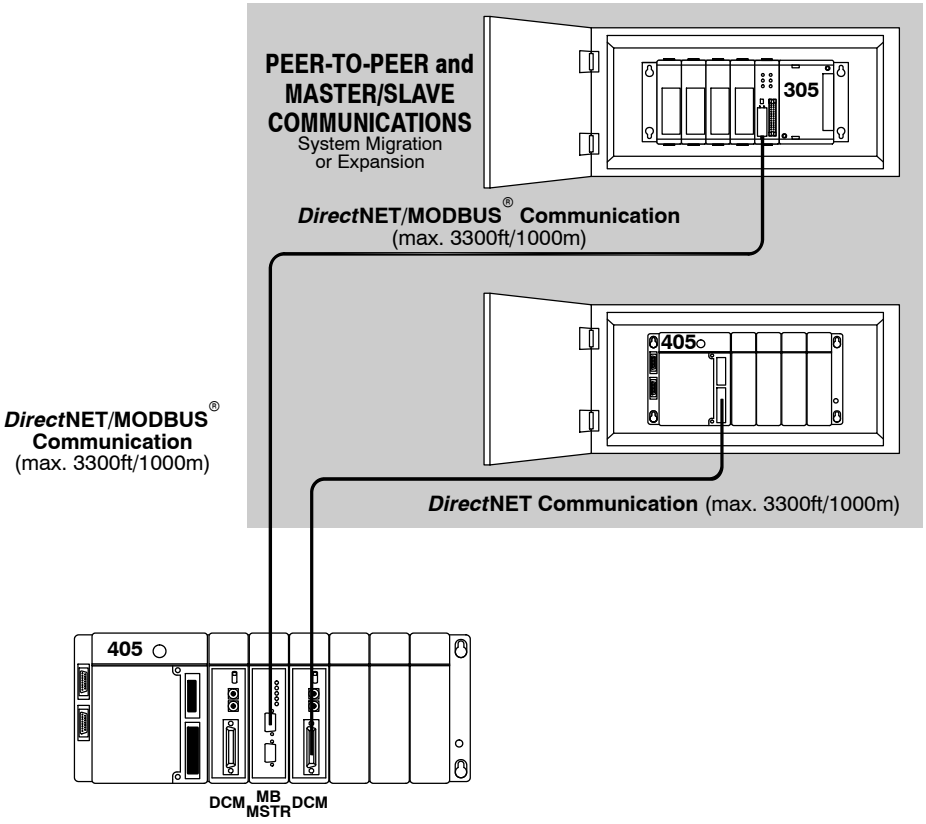
A DL405 system can be developed using many different arrangements of these configurations. All I/O configurations use the standard complement of DL405 I/O modules and bases. Below is a brief description of each of these configurations. Examples of each configuration are discussed in detail later in this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



Networking Configurations

The DL405 PLCs offer the following four ways to add I/O modules to the system:

- **Data Communications Module** – connects a DL405 system to devices using the ***Direct***NET protocol, or connects as a slave to a MODBUS network.
- **DL450 Communications Ports** – the DL450 CPU has two extra (total of four) built-in comm ports. It allow 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 DL405 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 DL405 system for connecting it as a slave to a MODBUS network, using the RTU protocol.
- **TIWAY® Network Interface Module** – Interface to Texas Instruments and Siemens TIWAY networks by using this module as a slave.
- **Shared Data Network Module** – The Shared Data Network Module lets you make peer–to–peer connections between DL405 PLC systems.



Module Placement and Configuration

Valid Module/Unit Locations

The most commonly used I/O modules for the DL405 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 DL405 system. Remember that the power budget can limit the number of modules in a base (discussed later).

Module/Unit	Local CPU Base	Local Exp. Base	Remote Base
CPUs	CPU Slot Only		
Expansion Units		CPU Slot Only	
8/16/32pt DC Input Modules	✓	✓	✓
64pt DC Input Modules	✓ Note 1	✓ Note 1, 2	
AC Input Modules	✓	✓	✓
AC/DC Input Modules	✓	✓	✓
8/16/32pt DC Output Modules	✓	✓	✓
64pt DC Output Modules	✓ Note 1	✓ Note 1, 2	
AC Output Modules	✓	✓	✓
Relay Output Modules	✓	✓	✓
Analog Modules	✓	✓	✓
Remote I/O			
Remote Master	✓		
Remote Slave Unit			CPU Slot Only
Communications and Networking Modules	✓	✓ Note 2	
CoProcessor Modules	✓		
Specialty Modules			
Interrupt	DL430 – Slot 0 Only DL440 – Slots 0 & 1 DL450 – Slots 0 & 1		
High Speed Counter	✓	✓	
PID Module	✓		
SDS	✓		
4 Loop Temp. Controller	✓		
Input Simulator	✓	✓	✓
Filler	✓	✓	✓

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.

I/O Configuration Methods

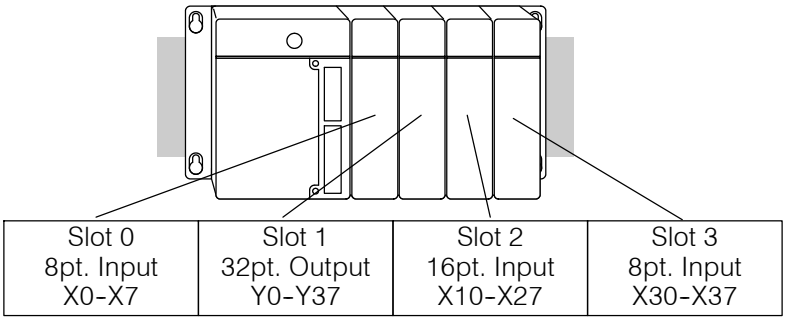
There are two methods of I/O configuration for the DL405 CPUs:

- **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** – (DL440/DL450 only) allows you 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 DL405 CPUs automatically detect any installed I/O modules (including specialty modules)at powerup, 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 system.



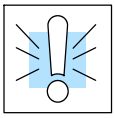
Both the Handheld Programmer and ***Direct***SOFT provide AUX functions that allow you to automatically configure the I/O. For example, with the Handheld Programmer AUX 46 executes an automatic configuration, which allows the CPU to examine the installed modules and determine the I/O configuration and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

Manual Configuration

✗	✓	✓
430	440	450

It may never become necessary, but DL440 and DL450 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 eight addresses will be unused.

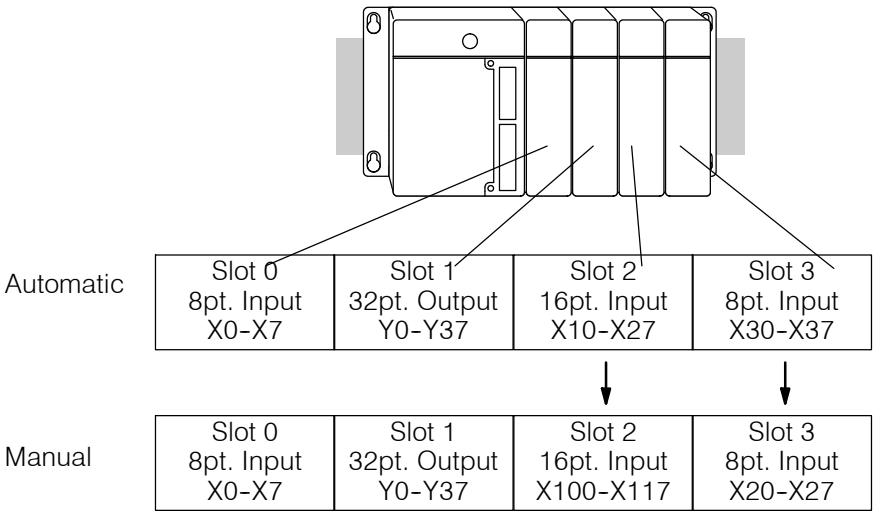


WARNING: If you manually configure an I/O slot, the I/O addressing for the other modules may change. This is because the DL405 CPUs do not allow you to assign duplicate I/O addresses. You must always correct any I/O configuration errors before you place the CPU in RUN mode. Uncorrected errors can cause 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.



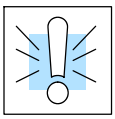
Power-On I/O Configuration Check

The DL405 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 Handheld Programmer or ***Direct***SOFT screen (use AUX 44 on the HPP to enable the configuration check).

If the system detects a change in the I/O configuration at power-up, an error code E252 NEW I/O CONFIGURATION will be generated. You can use AUX 42 to determine the exact base and slot location where the change occurred.

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

When a configuration error is generated, you may actually want to use the new I/O configuration. For example, you may have intentionally changed an I/O module to use with a program change. You can use AUX 45 to select the new configuration, or, keep the existing configuration stored in memory.



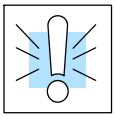
WARNING: Verify the I/O configuration being selected will work properly with the CPU program. Always correct any I/O configuration errors before placing the CPU in RUN mode. Uncorrected errors can cause unpredictable machine operation that can result in a risk of personal injury or damage to equipment.

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 DL405 system it is important to remember there is a 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 chose 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: It is *extremely* important to calculate the power budget correctly. If you exceed the power budget, the system may operate in an unpredictable manner which may result in a risk of personal injury or equipment damage.

CPU Power Specifications

The following chart shows the amount of current **available** for the two voltages supplied on the DL405 CPU, Expansion unit or Remote Slave unit. Use these currents when calculating the power budget for you system. The Auxiliary 24V Power Source mentioned in the table is a connection at the base terminal strip allowing you to connect to devices or DL405 modules that require 24VDC.

CPUs	5V Current Supplied in mA.	Auxiliary 24V Power Source Current Supplied in mA.		Remote and Expansion Units	5V Current Supplied in mA.	Auxiliary 24V Power Source Current Supplied in mA.
D4-430	3700	400		D4-EX	4000	400
D4-440	3700	400		D4-EXDC	4000	None
D4-440DC-1	3700	None		D4-EXDC-2	3700	None
D4-440DC-2	3700	None		D4-RS	3700	400
D4-450	3100	400		D4-RSDC	3700	None
D4-450DC-1	3100	None		H4-EBC	3470	400
D4-450DC-2	3100	None		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.

See Errata Sheet at the beginning of this file. These two current values have been revised.

Device	5V Current Required (mA)	External 24V Current Req. (mA)
I/O Bases		
D4-04B, D4-04BNX, D4-04B-1	80	None
D4-06B, D4-06BNX, D4-06B-1	80	None
D4-08B, D4-08BNX, 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
D4-32ND3-2	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
F4-08NES	90	None
DC Output Modules		
D4-08TD1	150	35
F4-08TD1S	295	None
D4-16TD1	200	125
D4-16TD2	400	None
D4-32TD1	250	140
D4-32TD1-1	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		
D4-HPP	320	None
DV-1000	150	None

Device	5V Current Required (mA)	External 24V Current Req. (mA)
Analog Modules		
F4-04AD	~~85~~ 150	100
F4-04ADS	~~270~~ 370	120
F4-08AD	75	90
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		
D4-ERM	320	None
D4-ERM-F	450	None
D4-RM	300	None
Communications and Networking		
D4-DCM	500	None
H4-ECOM	530	None
H4-ECOM-F	670	None
H4-ECOM100	300	None
F4-MAS-MB	235	None
CoProcessors™		
F4-CP128	305	None
F4-CP512	235	None
F4-CP128-T	350	None
Specialty Modules		
D4-16SIM	150	None
D4-HSC	300	None
F4-16PID	160	None
F4-8MPI	225	170
F4-4LTC	280	75
H4-CTRIO	400	None

Power Budget Calculation Example

The following example shows how to calculate the power budget for the DL405 system.

Base # 0	**Module Type**	**5 VDC (mA)**	**Auxiliary Power Source 24 VDC Output (mA)**
CPU/ Expansion Unit/ Remote Slave Used	D4-430	**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
Handheld Prog	D4-HPP	+ 320	+ 0
Maximum power required		**2770**	**155**
Remaining Power Available		3700-2950 =**930**	400 - 300 = **100**

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 24 VDC output. Pay special attention to the current supplied by either the CPU, Expansion Unit, and Remote Slave since they do differ. Devices which fall into the "**Other**" category are devices such as the Base and the Handheld programmer which also have power requirements but do not directly plug into the base.
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 used 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 # _____	Module Type	5 VDC (mA)	Auxiliary Power Source 24 VDC 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 24 VDC output. Pay special attention to the current supplied by either the CPU, Expansion Unit, and Remote Slave since they do differ. Devices which fall into the "**Other**" category are devices such as the Base and the Handheld programmer which also have power requirements but do not directly plug into the base.
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 used this configuration and you will need to restructure your I/O configuration.

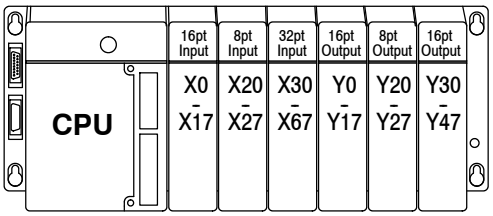
Local I/O Expansion

✓ ✓ ✓
430 440 450

The following I/O base configurations will assist you in understanding the options available in the DL405 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 base will use 256 I/O points . The status of each I/O point is updated each I/O scan of the CPU.

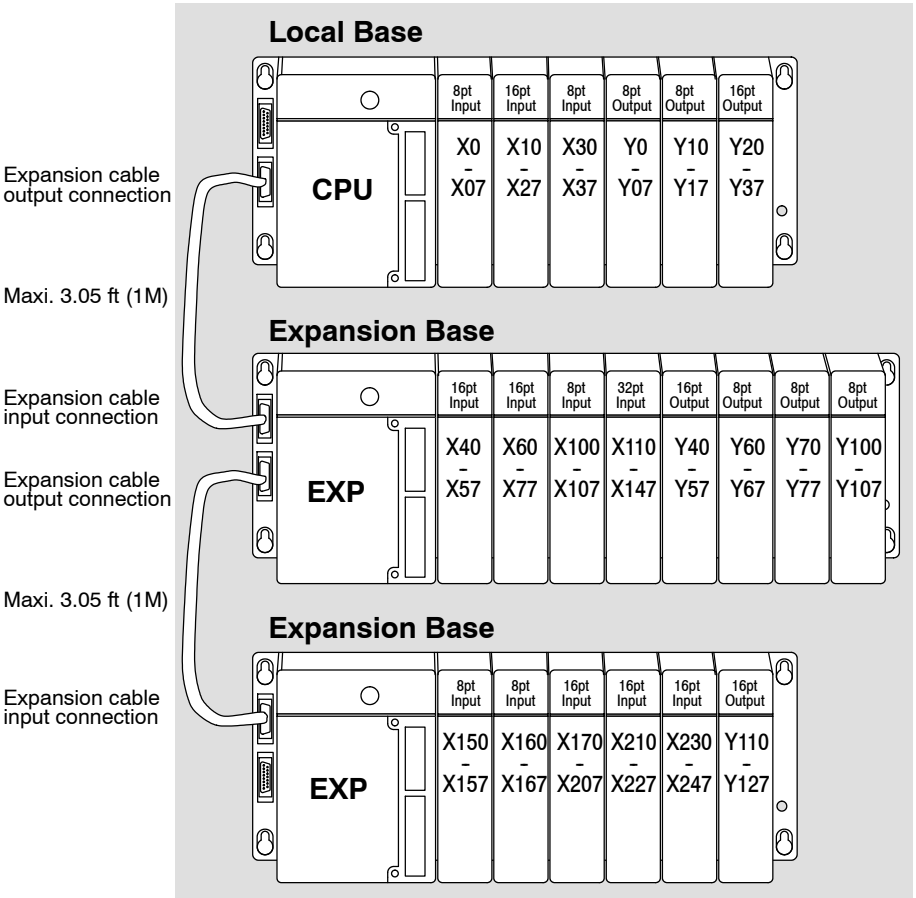


Local Expansion Base and I/O

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.

DL430/440: supports a maximum of 3 expansion bases, and maximum of 320 input points and 320 output points (includes local base I/O)

DL450: 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

✓ ✓ ✓
430 440 450

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:

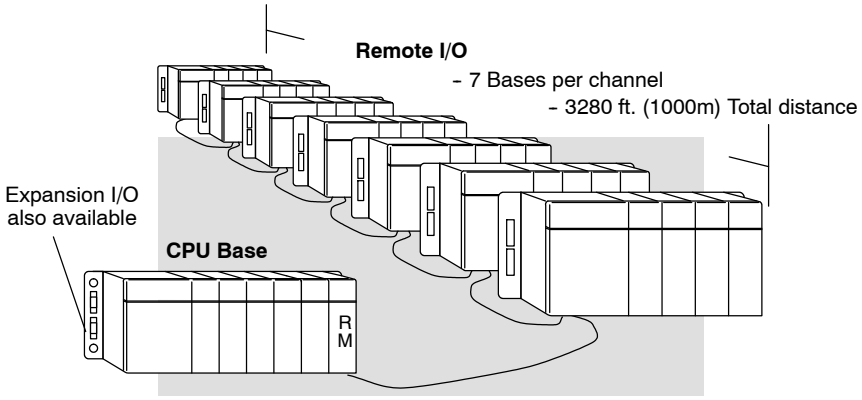
- **DL430** / **DL440 CPUs**: Remote I/O requires a remote master module (D4-RM) to be installed in the local CPU base. The CPU updates the remote master, then the remote master handles all communication to and from the remote I/O base by communicating to the remote slave module (D4-RS) installed in each remote base.
- **DL450 CPU**: The CPU's comm port 3 features a built-in Remote I/O channel. You may also use one or two D4-RM remote masters in the local base as described above (can use either or both methods).

	DL430	DL440	DL450
Maximum number of Remote Masters supported in the local CPU base (1 channel per Remote Master)	2	2	2
CPU built-in Remote I/O channels	none	none	1
Maximum I/O points supported by each channel	512	512	512
Maximum Remote I/O points supported	512	1024	1536
Maximum number of remote I/O bases per channel	7	7	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 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 with seven remote bases. If the CPU is a DL450, adding the first remote I/O channel does not require installing a remote master module (we use the CPU's built-in remote I/O channel on port 3).



Configuring the CPU's Remote I/O Channel

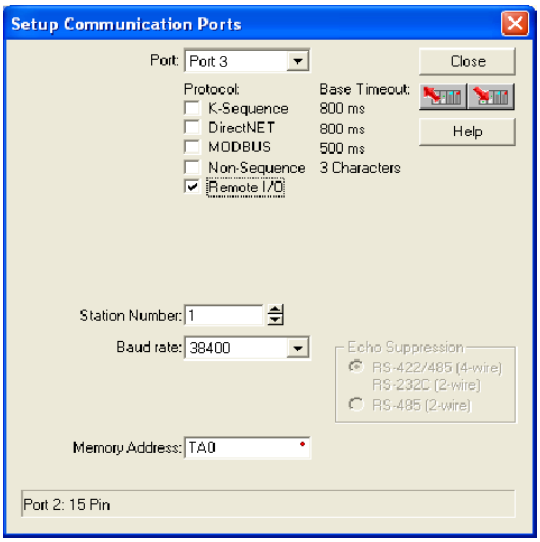
430	440	450
✗	✗	✓

This section describes how to configure the DL450's 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 manual exclusively when using regular Remote Masters and Remote Slaves for remote I/O in any DL405 system.

The DL450 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 DL450's Port 3 is capable of several protocols. To configure the port using the Handheld Programmer, use AUX 56 and follow the prompts, making the same choices as indicated below on this page. To configure the port in ***Direct***SOFT, choose the PLC menu, then **Setup > 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 it (called "M-NET" on the HPP). The dialog shown below will appear.



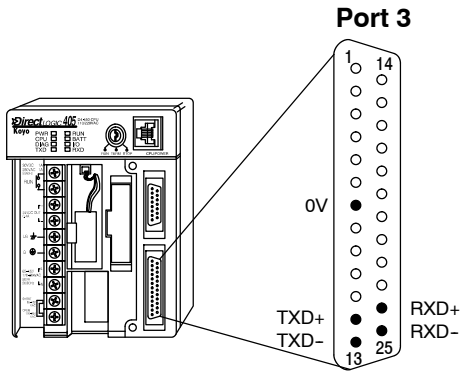
- **Station Number:** Choose "0" as the station number, which makes the DL450 the master. Station numbers 1-7 are reserved for remote slaves.
- **Baud Rate:** The baud rates 19200 and 38400 baud are available. Choose 38400 initially as the remote I/O baud rate, and revert to 19200 baud if you experience 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 a Remote I/O configuration table (V37700 is the default). This table is separate and independent from the table for any Remote Master(s) in the system.

Then click the button indicated to send the Port 3 configuration to the CPU, and click Close.

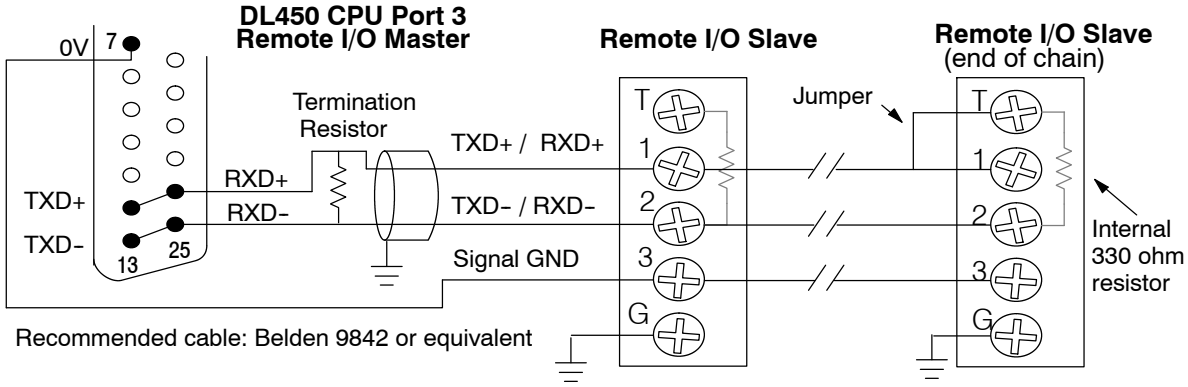
The next step is to make the connections between all devices on the Remote I/O link.

The location of the Port 3 on the DL450 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-



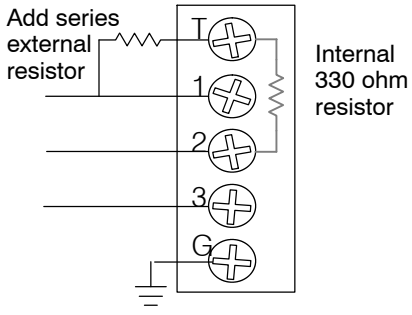
Now we are ready to discuss wiring the DL450 to the remote slaves on the remote base(s). The remote I/O link is a 3-wire, half-duplex type. Since Port 3 of the DL450 CPU is a 5-wire full duplex-capable port, we must jumper its transmit and receive lines together as shown below (converts it to 3-wire, half-duplex).



The twisted/shielded pair connects to the DL450 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.

Ideally, the two termination resistors at the cables opposite ends and the cable's rated impedance should match. For cable impedances greater than 330 ohms, add a series resistor at the last slave as shown to the right. If less than 330 ohms, just parallel a matching resistance across the slave's pins 1 and 2 instead. For example, to match the termination resistance to Belden 9842, use a 120 ohm resistor across terminals 1 and 2.

Remember to size the termination resistor at Port 3 to match. *The resistance values should be between 100 and 500 ohms.*



Configure Remote I/O Slaves

After configuring the DL450 CPU's 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 CPU's 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 DL450 CPU's 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.

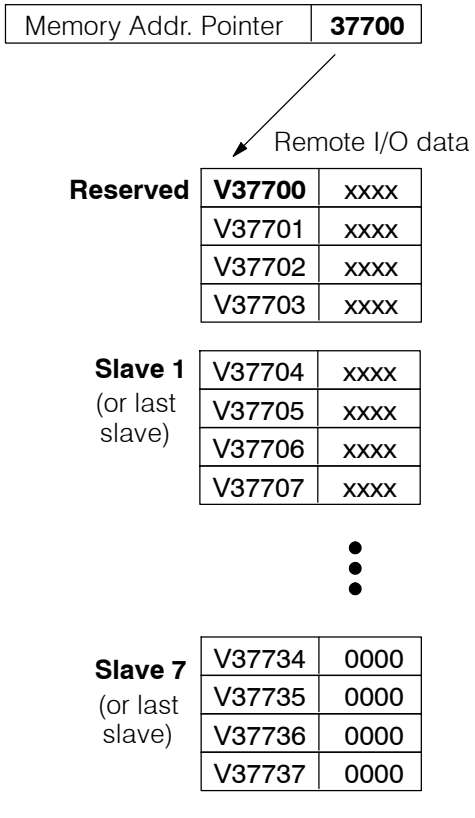
The CPU reads data from the table just after powerup, interpreting the four data words in each block with these meanings:

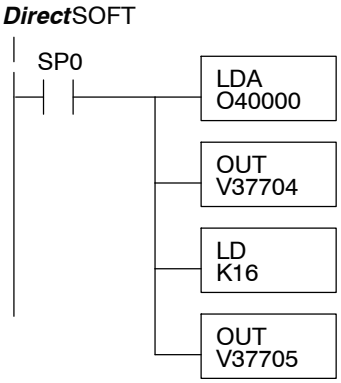
1. Starting address of slave's input data
2. Number of slave's input points
3. Starting address of outputs in slave
4. Number of slave's output points

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 powerup. Use the LDA instruction as shown to the right, to load an address to place in the table. Use the regular LD constant to load the number of the slave's input or output points.

The 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 give a shorter program example for one slave.

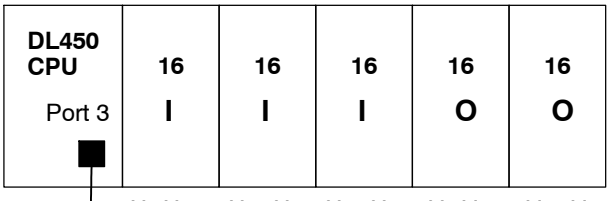




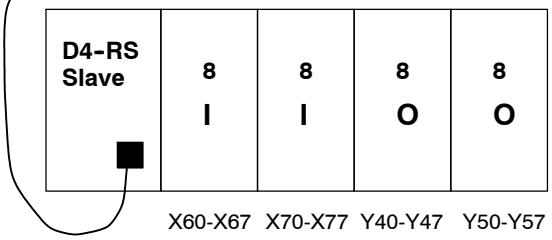
Consider the simple system featuring Remote I/O shown below. The DL450's built-in Remote I/O channel connects to one slave base, which we will assign a station address=1. The baud rates on the master and slave will be 38400 kB.

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 DL450, 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).

Main Base with CPU as Master



Remote Slave



Remote Slave Worksheet

Remote Base Address 1 **(Choose 1-7)**

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

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

Total Input Points 16

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

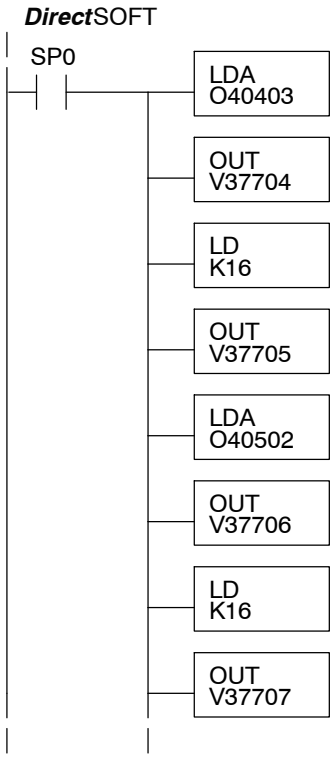
Total Output Points 16

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 Specifications and Operation.

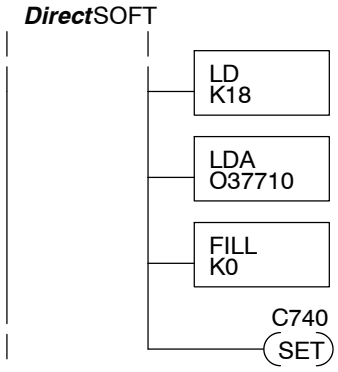
The program segment required to transfer our worksheet results to the Remote I/O configuration table is shown to the right. Remember to use the LDA or LD instructions appropriately.

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



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

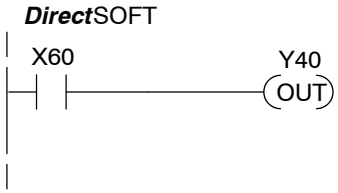
We continue our setup program from the previous page by adding a segment which fills the remainder of the table with zeros. The easiest way is the use the fill command as shown. The example to the right fills zeros for slave numbers 2–7, which do not exist in our example system (6 bases x 4 = 24 locations, = 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 *Direct*NET

Configuring the CPU's Comm Ports

✓ ✓ ✓
430 440 450

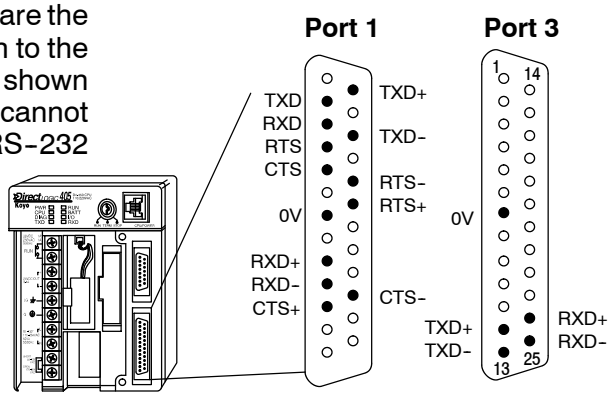
This section describes how to configure the CPU's built-in networking ports. for either MODBUS or ***Direct***NET. This will allow you to connect the DL405 PLC system directly to MODBUS networks using the RTU protocol, or to other devices on a ***Direct***NET network. MODBUS hosts system on the network must be capable of issuing the MODBUS commands to read or write the appropriate data. For details on the MODBUS protocol, please refer to the Gould MODBUS Protocol reference Guide (P1-MBUS-300 Rev. B). In the event a more recent version is available, check with your MODBUS supplier before ordering the documentation. For more details on ***Direct***NET, order our ***Direct***NET manual, part number DA-DNET-M.

NOTE: For information about the MODBUS protocol see the Group Schneider website at: www.schneiderautomation.com. At the main menu, select Support/Services, Modbus Technical Manuals, PI-MBUS-300 Modbus Protocol Reference Guide or search for PIMBUS300. For more information about ***Direct***NET protocol, order our ***Direct***NET user manual, part number DA-DNET-M, or download it free from our website: www.automationdirect.com. Select **Manuals/Docs > Online User Manuals > Misc. > DA-DNET-M**.

The DL430 and DL440 can be DirectNET slaves on port 1. Both the DL450's Port 1 and Port 3 can operate as master or slave for both MODBUS and ***Direct***NET. Port 1 has RS-232 and RS-422 signal levels available on separate pins, and Port 3 (DL450) uses RS-422 signal levels.

Ports 1 and Port 3 on the DL450 share the 25-pin D-shell connector, as shown to the right. Connect one or both ports as shown below. Note that you cannot simultaneously use Port 1's RS-232 signals and its RS-422 signals.

	Port 1	Port 3
DL430 and DL440	***Direct***NET, slave only	N/A
DL450	***Direct***NET or MODBUS, master/slave	***Direct***NET or MODBUS, master/slave

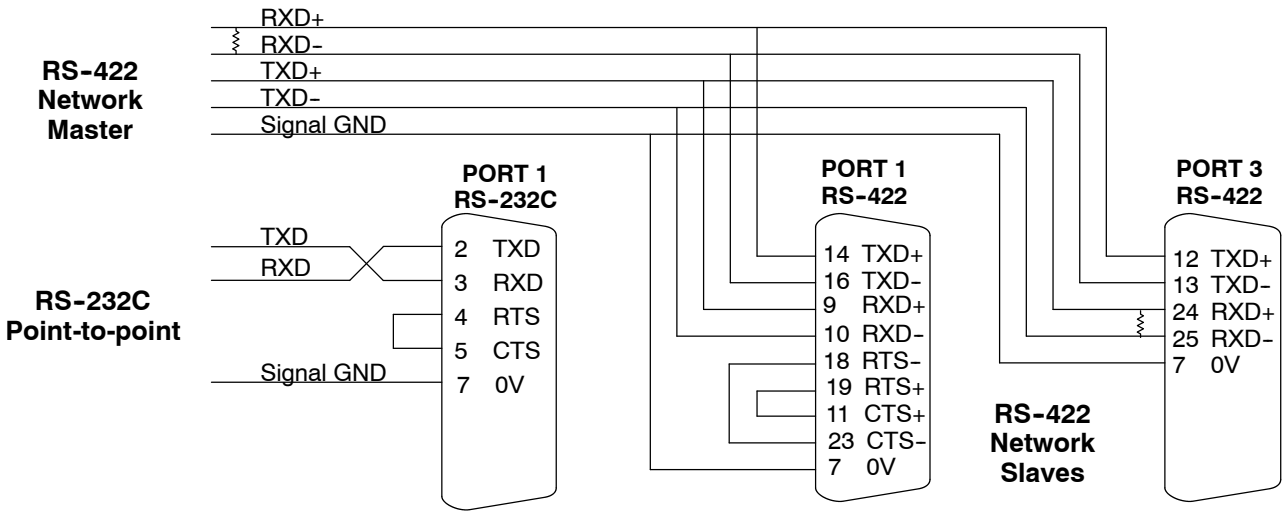


NOTE: The recommended cable for RS-232 or RS-422 is Belden 8102 or equivalent.

Change NOTE to say: "The recommended cable for RS-232 and RS-422 is AutomationDirect L19772-1 (Belden 8102) or equivalent."

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 your DL405 is to be used as a MODBUS Master and the distance will be more than 1000 feet, you can use the MODBUS Network Master module, F4-MAS-MB, and use the RS-485 port. See the module on our website, www.automationdirect.com for more details.

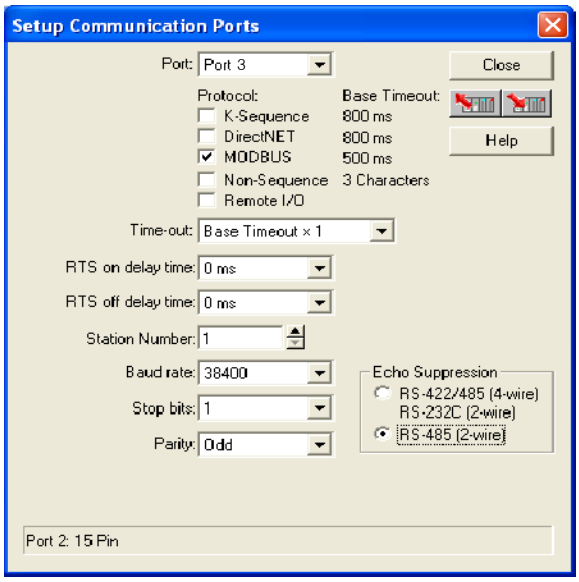


MODBUS Port Configuration

430	440	450
X	X	✓

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

- **Port:** From the port number list box at the top, choose Port 1 or 3.
- **Protocol:** Click the box to the left of MODBUS to select it (use AUX 56 on the HPP, and select MBUS). The dialog below will appear.



- **Timeout:** amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS on delay time:** the amount of time the port waits to send a message after it's ready to send. For port 1, it activates the RTS line before it begins transmitting (assuming CTS is already active). The port will not transmit if the CTS input is false.
- **Station Number:** For making the CPU port a MODBUS® master, choose "1". The possible range for MODBUS slave numbers is from 1 to 247, but the DL450 network instructions will access only slaves 1 to 90. Each slave must have a unique number. At powerup, the port is automatically a slave, unless and until the DL450 executes ladder logic network instructions which use the port as a master. Thereafter, the port reverts back to slave mode until ladder logic uses the port again.
- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value. Refer to the appropriate product manual for details.
- **Stop Bits:** Choose 1 or 2 stop bits for use in the protocol.
- **Parity:** Choose none, even, or odd parity for error checking.

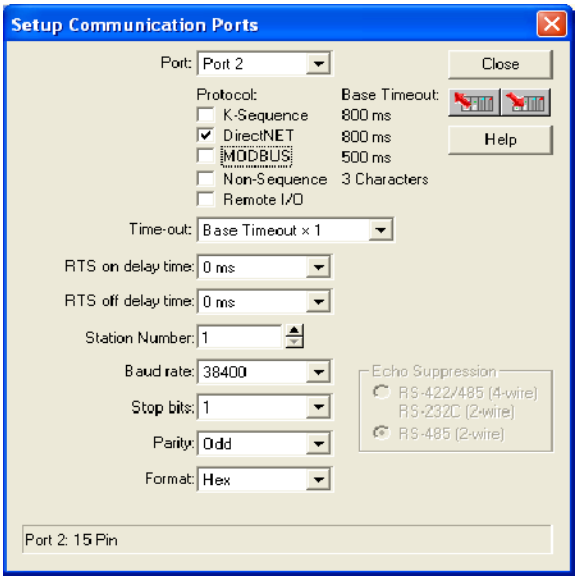
Then click the button indicated to send the Port configuration to the CPU, and click **Close**.

*Direct*NET Port Configuration

✓ ✓ ✓
430 440 450

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

- **Port:** From the port number list box, choose Port 1 or 3 (DL450 only).
- **Protocol:** Click the box to the left of DirectNET to select it (use AUX 56 on the HPP, then select DNET). The dialog below will appear.



- **Timeout:** amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS delay time:** the amount of time the port waits to send a message after it's ready to send. For port 1, it activates the RTS line before it begins transmitting (assuming CTS is already active). The port will not transmit if the CTS input is false.
- **Station Number:** For making the CPU port a ***Direct***NET master, choose "1". The allowable range for ***Direct***NET slaves is from 1 to 90 (each slave must have a unique number). At powerup, the port is automatically a slave, unless and until the DL450 executes ladder logic instructions which attempt to use the port as a master. Thereafter, the port reverts back to slave mode until ladder logic uses the port again.
- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value.
- **Stop Bits:** Choose 1 or 2 stop bits for use in the protocol.
- **Parity:** Choose none, even, or odd parity for error checking.
- **Format:** Choose between hex or ASCII formats.

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

Network Slave Operation

✓ ✓ ✓
430 440 450

This section describes how other devices on a network can communicate with a CPU port that you have configured as a ***Direct***NETslave or MODBUS slave (DL450). A MODBUS host must use the MODBUS RTU protocol to communicate with the DL450 as a slave. The host software must send a MODBUS function code and MODBUS address to specify a PLC memory location the DL450 comprehends. The ***Direct***NET host just uses normal I/O addresses to access any DL405 CPU and system. No CPU ladder logic is required to support either MODBUS slave or ***Direct***NET slave operation.

MODBUS Function Codes Supported

✗ ✗ ✓
430 440 450

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 DL450 supports the MODBUS function codes described below.

MODBUS Function Code	Function	DL405 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 DL405 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 type corresponds to any desired PLC memory location by using the cross-reference table below.

DL450 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 Value (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 type and address
- By specifying a MODBUS address only.

If Your Host Software Requires the Data Type and Address...

Many host software packages allow you to specify the MODBUS data type and the MODBUS address that 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 below shows the exact equation used for each group of data.

DL450 Memory Type	QTY (Dec.)	PLC Range (Octal)	MODBUS Address Range (Decimal)	MODBUS Data Type
For Discrete Data Types .... Convert PLC Addr. to Dec. + Start of Range +				**Data Type**
Inputs (X)	1024	X0 – X1777	2048 – 3071	Input
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 Addr. to Dec. +				**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

*** Note:** 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

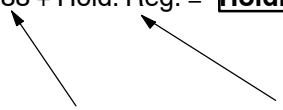
Find the MODBUS address for User V location V2100.

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

PLC Address (Dec.) + Data Type

V2100 = 1088 decimal

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



V-Memory, user data (V)	3072 12288	V1400 - V7377 V10000-V37777	768 - 3839 4096 - 16383	Holding Register

Example 2: Y20

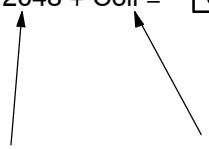
Find the MODBUS address for output Y20.

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

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

Y20 = 16 decimal

16 + 2048 + Coil = **Coil 2064**



Outputs (Y)	1024	Y0 - Y1777	2048 - 3071	Coil

Example 3: T10 Current Value

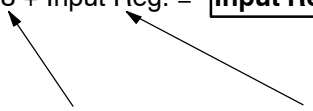
Find the MODBUS address to obtain the current value from Timer T10.

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 - V377	0 - 255	Input Register

Example 4: C54

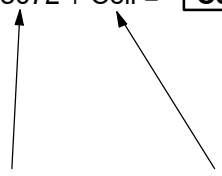
Find the MODBUS address for Control Relay C54.

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

C54 = 44 decimal

44 + 3072 + Coil = **Coil 3116**



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 the 584/984 mode allows access to a higher number of memory locations within each data type. If your software only supports 484 mode, then there may be some PLC memory locations that will be unavailable. The actual equation used to calculate the address depends on the type of PLC data you are using. The PLC memory types are split into two categories for this purpose.

- Discrete – X, GX, SP, Y, CR, S, T (contacts), C (contacts)
- Word – V, Timer current value, Counter current value

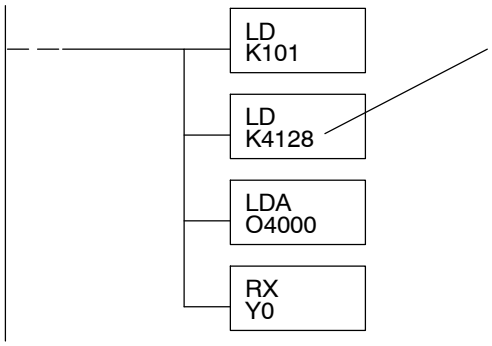
In either case, you basically just convert the PLC octal address to decimal and add the appropriate MODBUS addresses (as required). The table below shows the exact equation used for each group of data.

DISCRETE DATA TYPES				
Memory Type	**PLC Range (Octal)**	**Address (484 Mode)**	**Address (584/984 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 (New feature)

The DL450 will support **function 04** read input register **(Address 30001)**. To use function 04, put the number '4' into the most significant position (4xxx). Four digits must be entered for the instruction to work properly with this mode.



The Maximum constant possible is 4128. This is due to the 128 maximum number of Bytes that the RX/WX instruction can allow. The value of 4 in the most significant position of the word will cause the RX to use function 04 (30001 range).

1. Refer to the Memory Mapping section of this manual for the correct memory mapping size. Some of the addresses shown above might not pertain to your CPU.

2. For an automated MODBUS/Koyo address conversion utility, download the file **modbus_conversion.xls** from our website, **www.automationdirect.com**.

Example 1: V2100 584/984 Mode

Find the MODBUS address for User V location V2100.

1. Find V-memory in the table.
2. Convert V2100 into decimal (1088).
3. Add the MODBUS starting address for the mode (40001).

PLC Address (Dec.) + Mode Address

V2100 = 1088 decimal

1088 + 40001 = **41089**

For Word Data Types	PLC Address (Dec.)		+	Appropriate Mode Address		
Timer Current Values (V)	128	V0 – V177	0 – 127	3001	30001	Input Reg
Counter Current Values (V)	128	V1000 – V1177	512 – 639	3001	30001	Input Reg
V Memory, user data (V)	1024	V2000 – V3777	1024 – 2047	4001	40001	Hold Reg.

Example 2: Y20 584/984 Mode

Find the MODBUS address for output Y20.

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

PLC Addr. (Dec) + Start Address + Mode

Y20 = 16 decimal

16 + 2048 + 1 = **2065**

Outputs (Y)	320	Y0 – Y477	2048 – 2367	1	1	Coil
Control Relays (CR)	256	C0 – C377	3072 – 3551	1	1	Coil
Timer Contacts (T)	128	T0 – T177	6144 – 6271	1	1	Coil

Example 3: T10 Current Value 484 Mode

Find the MODBUS address to obtain the current value from Timer T10.

1. Find Timer Current Values in the table.
2. Convert T10 into decimal (8).
3. Add the MODBUS starting address for the mode (3001).

PLC Address (Dec.) + Mode Address

TA10 = 8 decimal

8 + 3001 = **3009**

For Word Data Types	PLC Address (Dec.)		+	Appropriate Mode Address		
Timer Current Values (V)	128	V0 – V177	0 – 127	3001	30001	Input Reg
Counter Current Values (V)	128	V1000 – V1177	512 – 639	3001	30001	Input Reg
V Memory, user data (V)	1024	V2000 – V3777	1024 – 2047	4001	40001	Hold Reg.

Example 4: C54 584/984 Mode

Find the MODBUS address for Control Relay C54.

1. Find Control Relays in the table.
2. Convert C54 into decimal (44).
3. Add the starting address for the range (3072).
4. Add the MODBUS address for the mode (1).

PLC Addr. (Dec) + Start Address + Mode

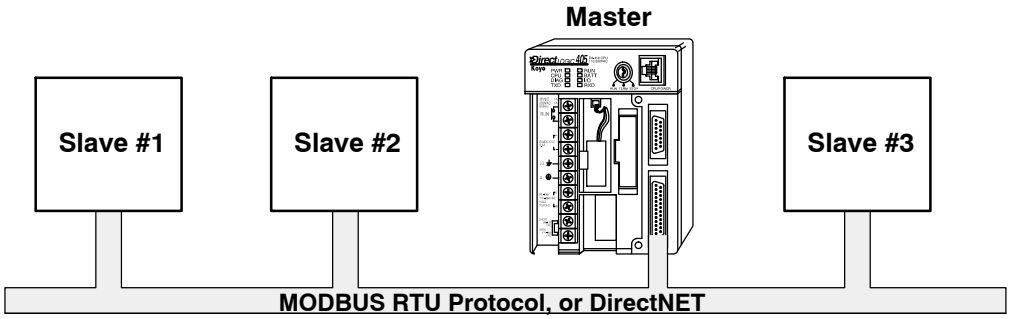
C54 = 44 decimal

44 + 3072 + 1 = **3117**

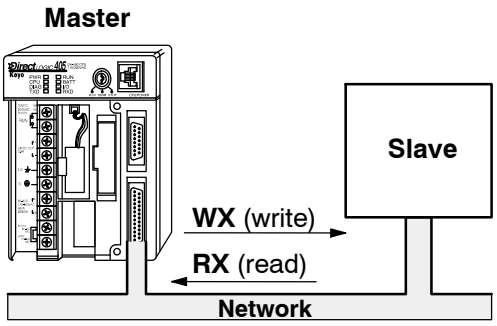
Outputs (Y)	320	Y0 – Y477	2048 – 2367	1	1	Coil
Control Relays (CR)	256	C0 – C377	3072 – 3551	1	1	Coil
Timer Contacts (T)	128	T0 – T177	6144 – 6271	1	1	Coil

Network Master Operation

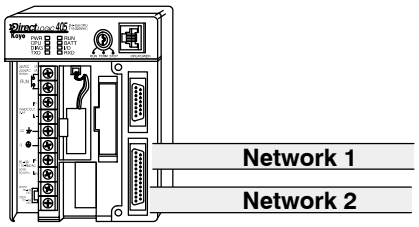
This section describes how the DL450 can communicate on a MODBUS or ***Direct***NET network as a master. For MODBUS networks, it uses the MODBUS RTU protocol, which must be interpreted by all the slaves on the network. Since MODBUS and ***Direct***NET are master / slave networks, the master station must initiate requests for network data transfers. This section teaches you how to design the required ladder logic for network master operation.



When using the DL450 CPU as the master station, you use simple RLL instructions to initiate the requests. The WX instruction initiates network write operations, and the RX instruction initiates network read operations. Before executing either the WX or RX commands, we will need to load data related to the read or write operation onto the CPU's accumulator stack. When the WX or RX instruction executes, it uses the information on the stack combined with data in the instruction box to completely define the task, which goes to the port.



It's possible to use both Port 1 and Port 3 for either MODBUS or ***Direct***NET, and to use either or both as masters. You must tell the WX and RX instructions the intended port for each communications transaction.

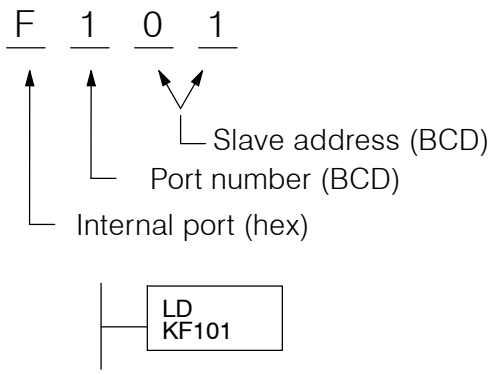


To summarize, the RLL instructions identify the following items.

1. Port number on the master (Port 1 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.

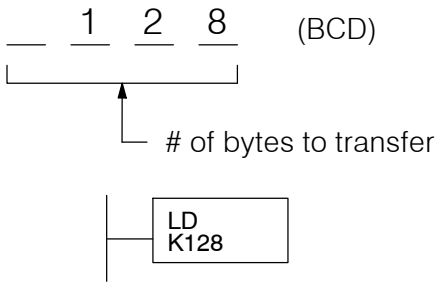
Step 1: Identify Master Port # and Slave #

The first Load (LD) instruction identifies the communications port number on the network master (DL450) and the address of the slave station. This instruction can address up to 90 MODBUS slaves, or 90 ***Direct***NET slaves. The format of the word is shown to the right. The "F" in the upper nibble tells the CPU the port is internal to the CPU (and not in a slot in the base). The second nibble indicates the port number, 1 or 3. The lower byte contains the slave address number in BCD (01 to 90).



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 DL405 Input points can be accessed by V-memory locations or as X input locations. However, if you only want X0 – X27, you'll have to use the X input data type because the V-memory locations can only be accessed in 2-byte increments. The following table shows the byte ranges for the various types of ***Direct***LOGIC™ products.

DL 205 / 405 Memory	Bits per unit	Bytes
V–memory	16	2
T / C current value	16	2
Inputs (X, GX, SP)	8	1
Outputs (Y, C, Stage, T/C bits)	8	1
Scratch Pad Memory	8	1
Diagnostic Status	8	1

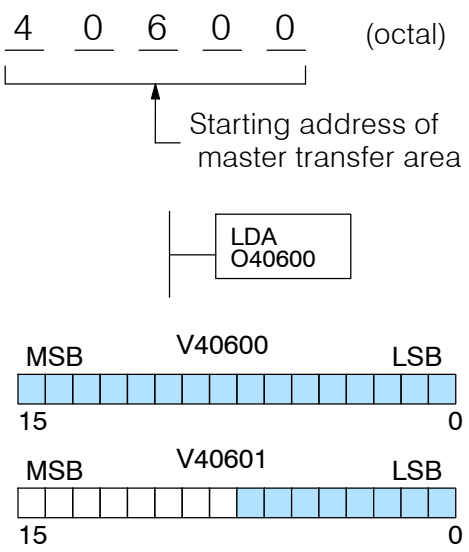
DL305 Memory	Bits per unit	Bytes
Data registers	8	1
T / C accumulator	16	2
I/O, internal relays, shift register bits, T/C bits, stage bits	1	1
Scratch Pad Memory	8	2
Diagnostic Status(5 word R/W)	16	10

Step 3: Specify Master Memory Area

The third instruction in the RX or WX sequence is a Load Address (LDA) instruction. Its purpose is to load the starting address of the memory area to be transferred. Entered as an octal number, the LDA instruction converts it to hex and places the result in the accumulator.

For a WX instruction, the DL450 CPU sends the number of bytes previously specified from its memory area beginning at the LDA address specified.

For an RX instruction, the DL450 CPU reads the number of bytes previously specified from the slave, placing the received data into its memory area beginning at the LDA address specified.

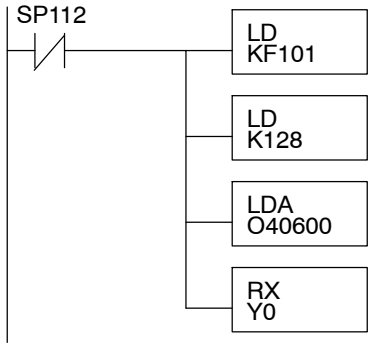


NOTE: Since V-memory words are always 16 bits, you may not always use the whole word. For example, if you only specify 3 bytes and you are reading Y outputs from the slave, you will only get 24 bits of data. In this case, only the 8 least significant bits of the last word location will be modified. The remaining 8 bits are not affected.

Step 4: Specify Slave Memory Area

The last instruction in our sequence is the WX or RX instruction itself. Use WX to write to the slave, and RX to read from the slave. All four of our instructions are shown to the right. In the last instruction, you must specify the starting address and a valid data type for the slave.

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.



- ***Direct***NET slaves – specify the same address in the WX and RX instruction as the slave's native I/O address
- MODBUS DL405 or DL205 slaves – specify the same address in the WX and RX instruction as the slave's native I/O address
- MODBUS 305 slaves – use the following table to convert DL305 addresses to MODBUS addresses

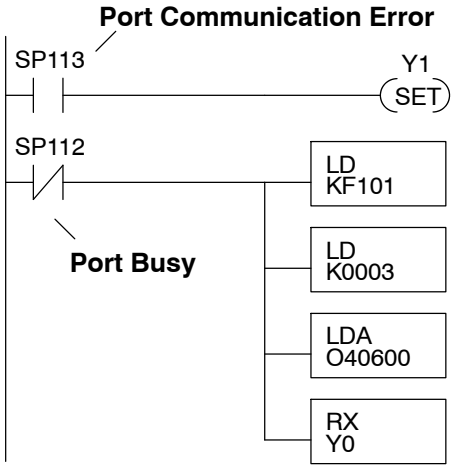
DL305 Series CPU Memory Type-to-DL405 Series CPU Memory

PLC Memory type	305 base address	405 base addr.	PLC Memory Type	305 base address	405 base addr.
TMR/CNT Current Values	R600	V0	TMR/CNT Status Bits	CT600	GY600
I/O Points	IO 000	GY0	Control Relays	CR160	GY160
Data Registers	R401, R400	V100	Shift Registers	SR400	GY400
Stage Status Bits (D3-330P only)	S0	GY200			

Communications from a Ladder Program

In some applications, the DL450 CPU as a network master will communicate only periodically to slaves(s) on the network. However, most applications will probably want to make a "continuous" update of memory areas from a slave to the master.

This normally means starting the task on each PLC SCAN. However, a single WX or RX network communication will probably last longer than one PLC scan time. *And we must wait before executing another RX or WX until the port has finished transmitting the previous WX or RX data.*



Each port which can be a master has two Special Relay contacts associated with it (see Appendix D for comm port special relays).One indicates "Port busy", and the other indicates "Port Communication Error". The example above shows the use of these contacts for a network master that only reads a device (RX). The Port Busy contact ensures one network transaction finishes before we begin another.

Use of the communication error SP relay is optional. If used, be sure to place it at the beginning of the communication routines, because a comm error relay is always reset (turned off) whenever an RX or WX instruction using the same port executes.

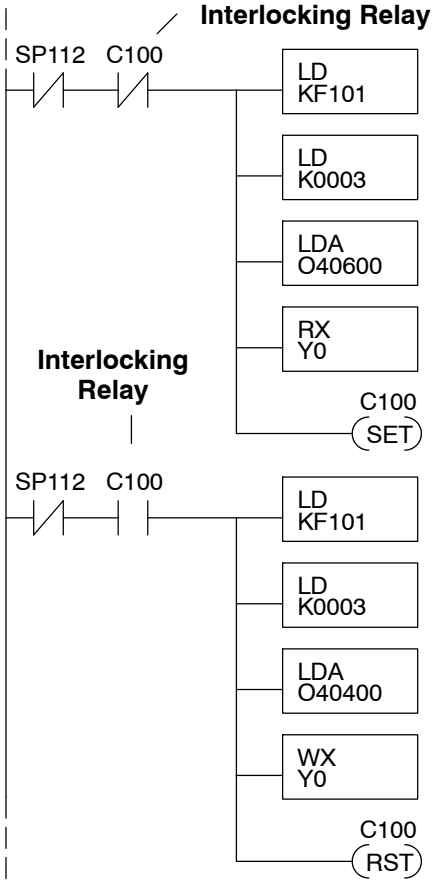
Multiple Read and Write Interlocks

If you're using multiple reads and writes in the RLL program, you have to interlock the routines to make sure all the routines are executed. If you don't use the interlocks, then the CPU will only execute the first routine. This is because each port can only handle one transaction at a time.

In the example to the right, after the RX instruction is executed, C0 is set. When the port has finished the communication task, the second routine is executed and C0 is reset.

If you're using RLL*PLUS* Stage Programing, you can just put each routine in a separate program stage to ensure proper execution. In most cases, RLL*PLUS* is a much more efficient way to create an automation program.

The ***Direct***NET manual provides a master/slave example with both RLL and Stage program descriptions (they are easily adapted for use with MODBUS).



5

Standard RLL Instructions

In This Chapter. . . .

— Introduction
— Boolean Instructions
— Comparative Boolean Instructions
— Immediate Instructions
— Timer, Counter, and Shift Register Instructions
— Accumulator / Data Stack and Output Instructions
— Accumulator Logic Instructions
— Math Instructions
— Bit Operation Instructions
— Number Conversion Instructions
— Table Instructions
— Clock / Calender Instructions
— CPU Control Instructions
— Program Control Instructions
— Interrupt Instructions
— Intelligent I/O Instructions
— Network Instructions
— Message Instructions

Introduction

The DL405 instruction set can perform many different types of operations. This chapter shows you how to use these individual instructions. The following table provides a quick reference listing of the instruction mnemonic and the page(s) defining the instruction. Each instruction definition will show in parentheses the HPP keystrokes used to enter the instruction. There are two ways to locate instructions:

- If you know the instruction category (Boolean, Comparative Boolean, etc.), just use the header at the top of the page to find the pages that discuss the instructions in that category.
- If you know the individual instruction mnemonic, use the following table.

The DL450 provides all of the instructions in the table, the DL440 provides a subset, and the DL430 a smaller subset. The instruction definitions indicate which CPUs feature the instruction. In Example 1, only the DL440 and DL450 have the instruction. In Example 2, all CPUs have the instruction.

Example 1

✗	✓	✓
430	440	450

Example 2

✓	✓	✓
430	440	450

If you are using a DL450 PLC (with firmware v3.30 or later) with *Direct*SOFT5 programming software, you can also use the IBox instructions covered in the DL405-IBOX-S Supplement.

Instruction	Page
ACON	5-196
ACOSR	5-119
ADD	5-87
ADDB	5-99
ADDBD	5-100
ADDBS	5-113
ADDD	5-88
ADDF	5-105
ADDR	5-89
ADDS	5-109
AND	5-14, 5-31, 5-70
ANDD	5-71
ANDND	5-23
ANDS	5-73
ANDSTR	5-16
ANDB	5-15
ANDD	5-71
ANDE	5-28
ANDF	5-72
ANDI	5-34
ANDMOV	5-171
ANDN	5-14, 5-31
ANDNB	5-15
ANDNE	5-28
ANDNI	5-34
ANDPD	5-23
ANDS	5-73

Instruction	Page
ANDSTR	5-16, 5-118
ASINR	5-118
ATANR	5-119
ATH	5-135
ATT	5-159
BCALL	7-27
BCD	5-129
BCDCPL	5-131
BEND	7-27
BIN	5-128
BLK	7-27
BREAK	5-178
BTOR	5-132
CMP	5-82
CMPD	5-83
CMPF	5-84
CMPR	5-86
CMPS	5-85
CNT	5-47
COSR	5-118
CV	7-25
CVJMP	7-25
DATE	5-175
DEC	5-117
DECB	5-120
DECO	5-127

Instruction	Page
DEGR	5-134
DISI	5-186
DIV	5-96
DIVB	5-104
DIVBS	5-116
DIVD	5-97
DIVF	5-108
DIVR	5-98
DIVS	5-112
DLBL	5-196
DRUM	6-15
EDRUM	6-18
ENCO	5-126
END	5-177
ENI	5-186
FAULT	5-195
FDGT	5-144
FILL	5-142
FIND	5-143
FINDB	5-173
FOR	5-180
GOTO	5-179
GRAY	5-139
GTS	5-181
HISTRY	5-197
HTA	5-136
INC	5-117

Instruction	Page
INCB	5–120
INT	5–185
INV	5–130
IRT	5–186
IRTC	5–186
ISG	7–24
JMP	7–24
LBL	5–179
LD	5–59
LDA	5–61
LDD	5–59
LDF	5–60
LDI	5–37
LDIF	5–38
LDLBL	5–162
LDR	5–64
LDSX	5–63
LDX	5–62
MDRMD	6–21
MDRMW	6–24
MLR	5–183
MLS	5–183
MOV	5–146
MOVMC	5–162
MUL	5–93
MULB	5–103
MULBS	5–115
MULD	5–94
MULF	5–107
MULR	5–95
MULS	5–111
NCON	5–197
NEXT	5–180
NJMP	7–24
NOP	5–177
NOT	5–19
OR	5–12, 5–30, 5–74
ORB	5–13
ORD	5–75
ORE	5–27
ORF	5–76
ORI	5–33
ORMOV	5–171
ORN	5–12, 5–30
ORNB	5–13

Instruction	Page
ORND	5–22
ORNE	5–27
ORNI	5–33
OROUT	5–19
OROUTI	5–35
ORPD	5–22
ORS	5–77
ORSTR	5–16
OUT	5–17, 5–65
OUTB	5–18
OUTD	5–65
OUTF	5–66
OUTI	5–35, 5–39
OUTIF	5–40
OUTL	5–68
OUTM	5–68
OUTX	5–67
PAUSE	5–20
PD	5–20
POP	5–69
PRINT	5–201
RADR	5–134
RD	5–189
RFB	5–150
RFT	5–156
ROTL	5–124
ROTR	5–125
RST	5–24
RSTB	5–25
RSTBIT	5–167
RSTI	5–36
RSTWT	5–178
RT	5–181
RTC	5–181
RTOB	5–133
RX	5–191
SBR	5–181
SEG	5–138
SET	5–24
SETB	5–25
SETBIT	5–167
SETI	5–36
SFLDGT	5–140
SG	7–23

Instruction	Page
SGCNT	5–49
SHFL	5–122
SHFR	5–123
SINR	5–118
SQRTR	5–119
SR	5–53
STOP	5–177
STR	5–10, 5–29
STRB	5–11
STRE	5–26
STRI	5–32
STRN	5–10, 5–29
STRNB	5–11
STRND	5–21
STRNE	5–26
STRNI	5–32
STRPD	5–21
STT	5–153
SUB	5–90
SUBB	5–101
SUBBD	5–102
SUBBS	5–114
SUBD	5–91
SUBF	5–106
SUBR	5–92
SUBS	5–110
SUM	5–121
SWAP	5–174
TANR	5–118
TIME	5–176
TMR	5–42
TMRA	5–44
TMRAF	5–44
TMRF	5–42
TSHFL	5–169
TSHFR	5–169
TTD	5–147
UDC	5–51
WT	5–190
WX	5–192
XOR	5–78
XORD	5–79
XORF	5–80
XORMOV	5–171
XORS	5–81

Using Boolean Instructions

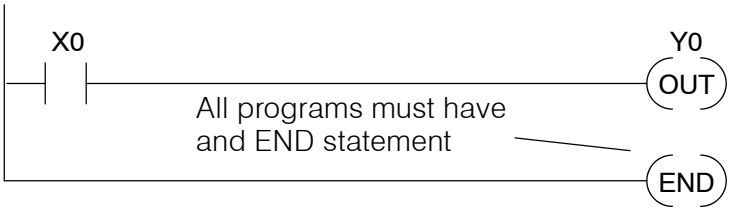
Do you ever wonder why so many PLC manufacturers always quote the scan time for a 1K boolean program? Most all programs utilize many Boolean instructions. These are typically very simple instructions designed to join input and output contacts in various series and parallel combinations. Our ***Direct***SOFT software is a similar program; therefore, you don't necessarily have to know the instruction mnemonics in order to develop your program. However, knowledge of mnemonics will be helpful whenever it becomes necessary to troubleshoot the program using a handheld programmer (HPP).

Many of the instructions in this chapter are not program instructions used in ***Direct***SOFT, but are implied. In other words, they are not actually keyboard commands, however, they can be seen in a **Mnemonic View** of the program once the ***Direct***SOFT program has been developed and accepted (compiled). Each instruction listed in this chapter will have a box to indicate how the instruction is used with ***Direct***SOFT and the HPP. The box will either contain **IMP** for implied, a ✓ for used or an ✕ for not used. The abbreviation, **DS**, or **HPP** will appear beneath the appropriate box as shown below.

IMP	✓	✕
DS	HPP	DS

END Statement

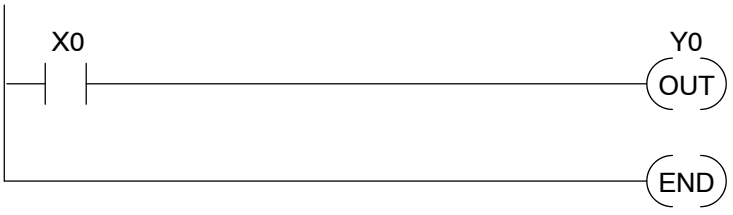
DL405 programs require an END statement (coil) as the last instruction. This tells the CPU this is the end of the program. Normally, any instructions placed after the END statement will not be executed. There are exceptions to this such as interrupt routines, etc. Chapter 5 discusses the instruction set in detail.



Simple Rungs

You use a contact to start rungs that contain both contacts and coils. The boolean instruction that does this is called a Store or, STR instruction. The output point is represented by the Output or, OUT instruction. The following example shows how to enter a single contact and a single output coil.

DirectSOFT Example



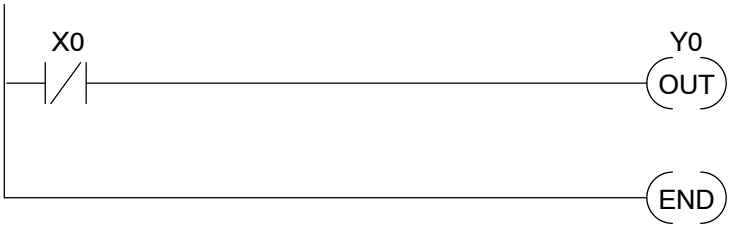
Handheld Mnemonics

```
STR X0
OUT Y0
END
```

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.

DirectSOFT Example



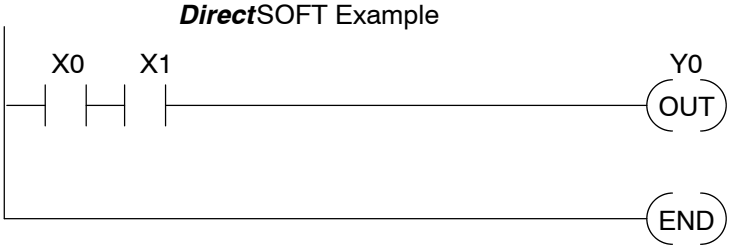
Handheld Mnemonics

```
STRN X0
OUT Y0
END
```

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.

DirectSOFT Example



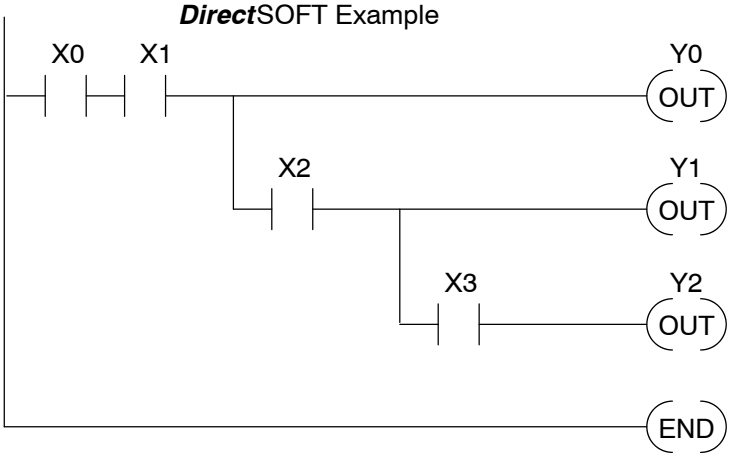
Handheld Mnemonics

```
STR X0
AND X1
OUT Y0
END
```

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.

DirectSOFT Example



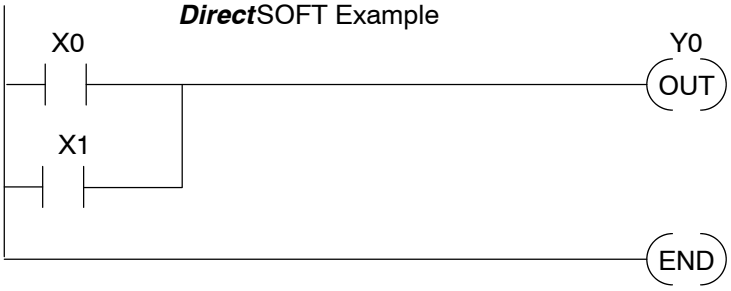
Handheld Mnemonics

```
STR X0
AND X1
OUT Y0
AND X2
OUT Y1
AND X3
OUT Y2
END
```

Parallel Elements

You 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.

DirectSOFT Example



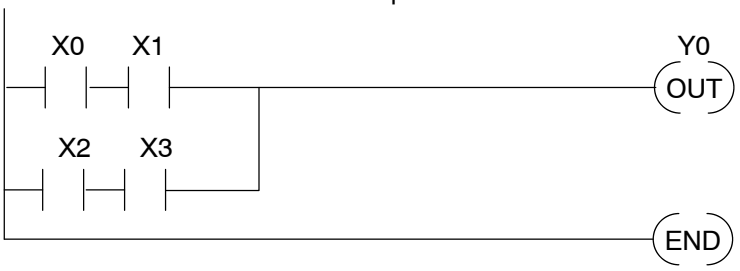
Handheld Mnemonics

```
STR X0
OR X1
OUT Y0
END
```

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.

DirectSOFT Example



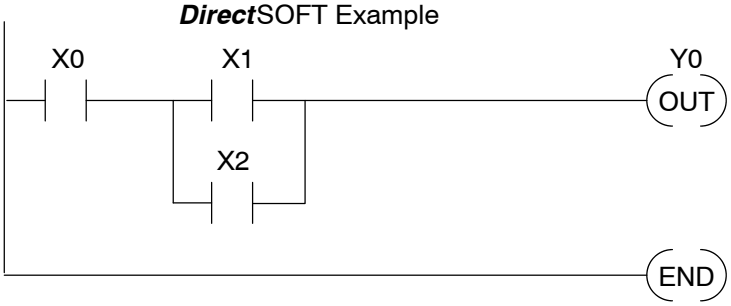
Handheld Mnemonics

```
STR X0
AND X1
STR X2
AND X3
ORSTR
OUT Y0
END
```

Joining Parallel Branches in Series

Quite often it is also necessary to 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.

DirectSOFT Example

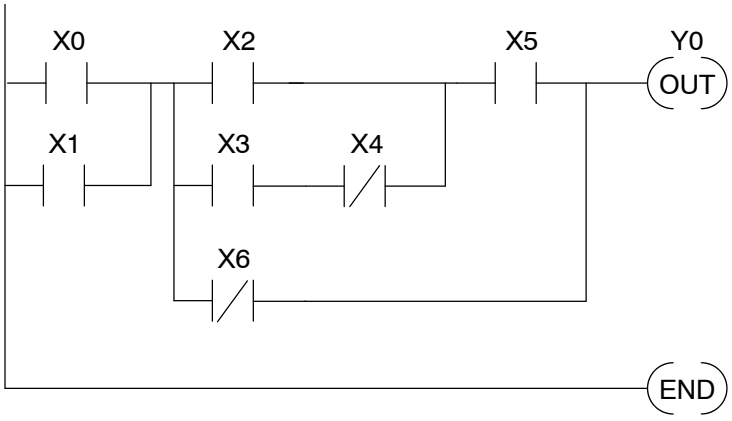


Handheld Mnemonics

```
STR X0
STR X1
OR X2
ANDSTR
OUT Y0
END
```

Combination Networks

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



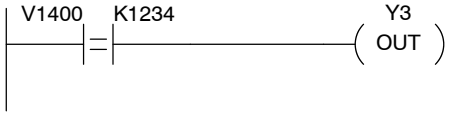
Handheld Mnemonics

```
STR X0
OR X1
STR X2
STR X3
ANDN X4
ORSTR
AND X5
ORN X6
ANDSTR
OUT Y0
```

Comparative Boolean

The Comparative Boolean evaluates two 4-digit BCD/hex values using boolean contacts. The valid evaluations are: equal to, not equal to, equal to or greater than, and less than.

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

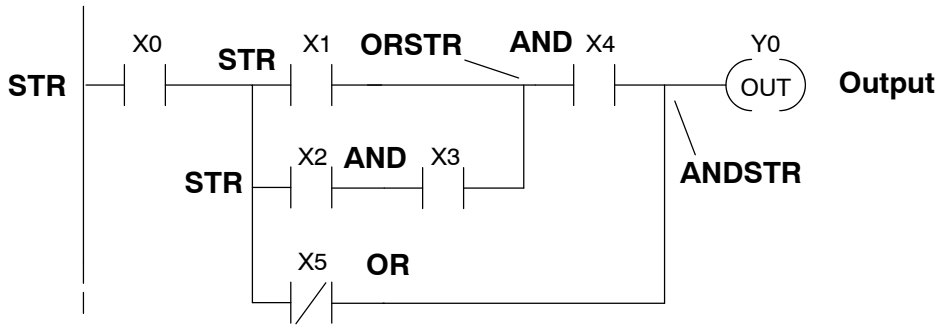


Boolean Stack

There are limits to how many elements you can include in a rung. This is because the DL405 CPUs use an 8-level boolean stack to evaluate the various logic elements. The boolean stack is a temporary storage area that solves the logic for the rung. Each time you enter a STR instruction, the instruction is placed on the top of the boolean stack. Any other STR instructions on the boolean stack are pushed down a level. The ANDSTR, and ORSTR instructions combine levels of the boolean stack when they are encountered. Since the boolean stack is only eight levels, an error will occur if the CPU encounters a rung that uses more than the eight levels of the boolean stack.

All of you software programmers may be saying, "I use ***Direct***SOFT, so I don't need to know how the stack works." Not quite true. Even though you can build the network with the graphic symbols, the limits of the CPU are still the same. If the stack limit is exceeded when the program is compiled, an error will occur.

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	

OR X5

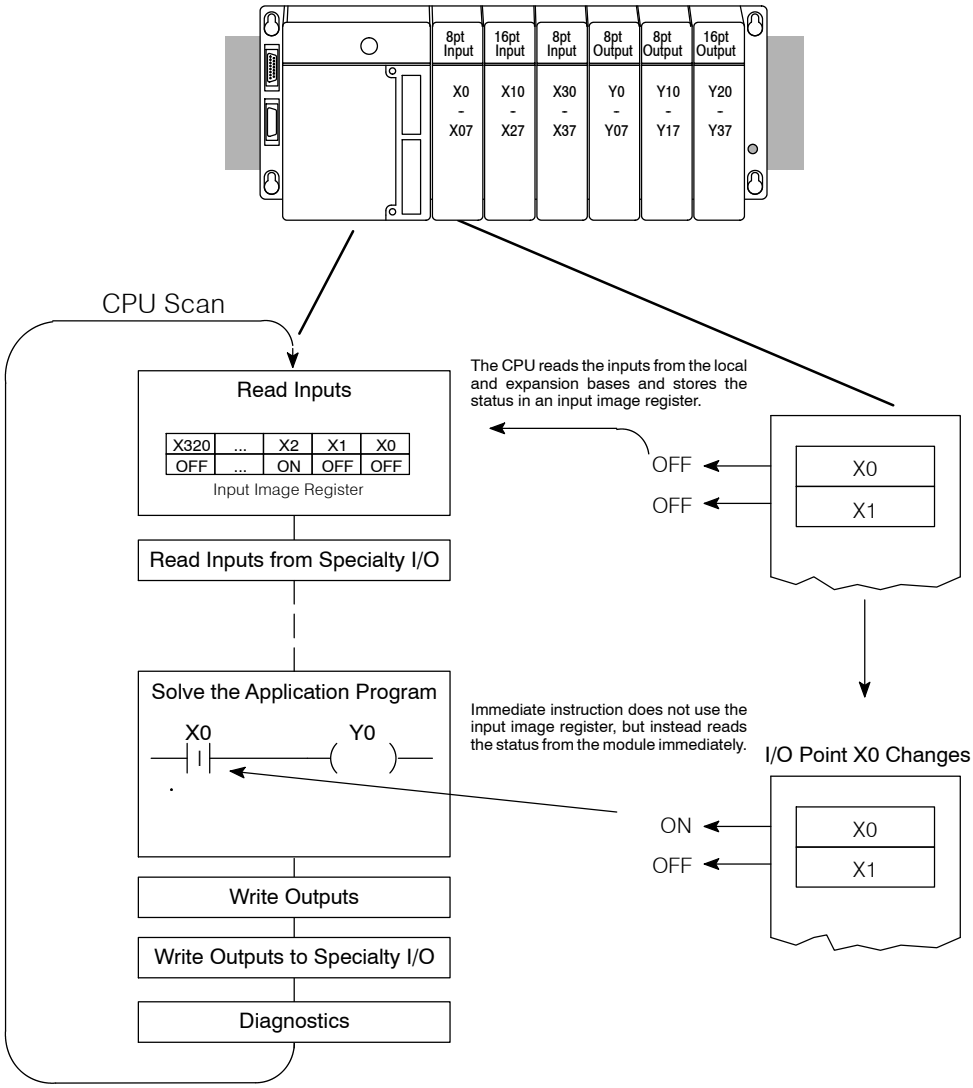
1	NOT X5 OR X4 AND [X1 OR (X2 AND X3)]
2	STR X0
3	

ANDSTR

1	X0 AND (NOT X5 OR X4) AND [X1 OR (X2 AND X3)]
2	
3	

Immediate Boolean The DL405 CPUs usually can 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 DL405 CPUs 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 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 module. 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 module, 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 module again to update the status. The immediate output instruction will write the status to the module and update the image register.



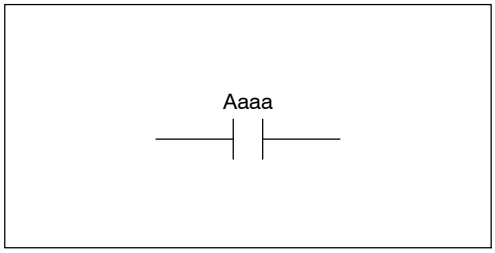
Boolean Instructions

Store (STR)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

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.

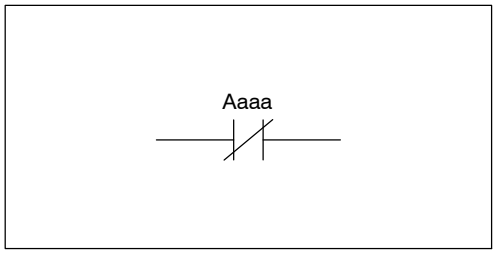


Store Not (STRN)

430	440	450
✓	✓	✓

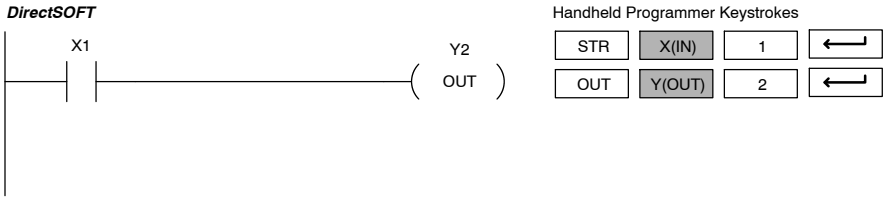
DS	HPP
✓	✓

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

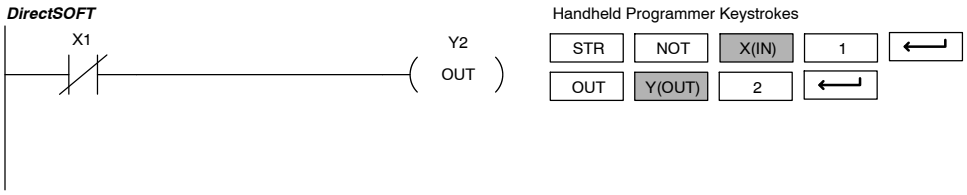


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
Inputs	X	0-477	0-477	0-1777
Outputs	Y	0-477	0-477	0-1777
Control Relays	C	0-737	0-1777	0-3777
Stage	S	0-577	0-1777	0-1777
Timer	T	0-177	0-377	0-377
Counter	CT	0-177	0-177	0-377
Special Relay	SP	0-137, 320-617	0-137 320-717	0-137, 320-717
Global	GX	0-777	0-1777	0-2777
Global	GY	-	-	0-2777

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.

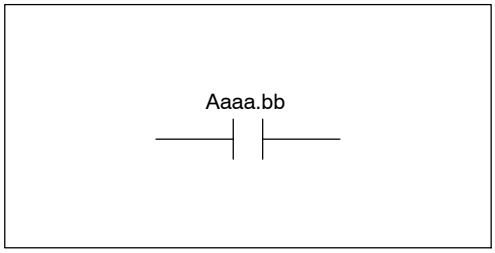


Store Bit-of-Word (STRB)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

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.

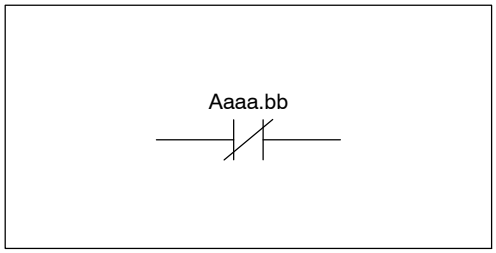


Store Not Bit-of-Word (STRNB)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

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



Operand Data Type		DL450 Range	
	A	aaa	bb
V-memory	B	All (See p. 3-42)	BCD, 0 to 15
Pointer	PB	All (See p. 3-42)	BCD, 0 to 15

In the following Store Bit-of-Word example, when bit 12 of V-memory location V1400 is on, output Y2 will energize.

DirectSOFT

Handheld Programmer Keystrokes

STR	SHFT	B	SHFT	V	1	4	0	0
K(con)	1	2	↵					
OUT	Y(OUT)	2	↵					

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

DirectSOFT

Handheld Programmer Keystrokes

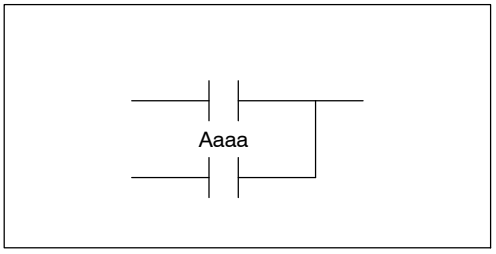
STR	STR	SHFT	B	SHFT	V	1	4	0	0
K(con)	1	2	↵						
OUT	Y(OUT)	2	↵						

Or (OR)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

The Or instruction logically ors a normally open contact in parallel with another contact in a rung. The status of the contact will be the same state as the associated image register point or memory location.

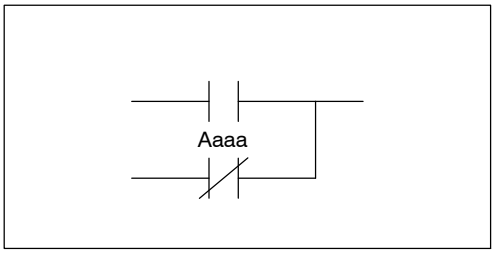


Or Not (ORN)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

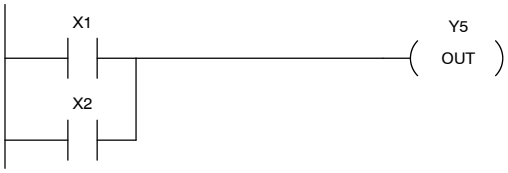
The Or Not instruction logically ors a normally closed contact in parallel with another contact in a rung. The status of the contact will be opposite the state of the associated image register point or memory location.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
Inputs	X	0-477	0-477	0-1777
Outputs	Y	0-477	0-477	0-1777
Control Relays	C	0-737	0-1777	0-3777
Stage	S	0-577	0-1777	0-1777
Timer	T	0-177	0-377	0-377
Counter	CT	0-177	0-177	0-377
Special Relay	SP	0-137, 320-617	0-137 320-717	0-137, 320-717
Global	GX	0-777	0-1777	0-1777
Global	GY	-	-	0-1777

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

DirectSOFT



Handheld Programmer Keystrokes

STR	X(IN)	1	↵
OR	X(IN)	2	↵
OUT	Y(OUT)	5	↵

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

DirectSOFT

Handheld Programmer Keystrokes

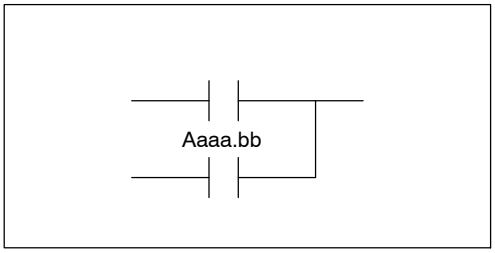
STR	X(IN)	1	↵	
OR	NOT	X(IN)	2	↵
OUT	Y(OUT)	5	↵	

Or Bit-of-Word (ORB)

430	440	450
X	X	✓

IMP	
DS	HPP
	✓

The Or Bit-of-Word instruction logically ors 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.

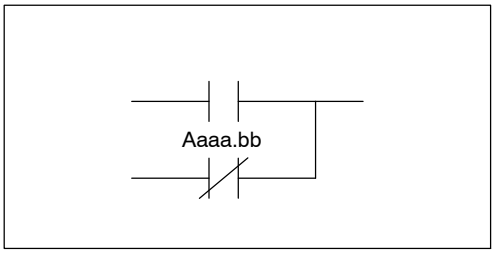


Or Not Bit-of-Word (ORNB)

430	440	450
X	X	✓

IMP	
DS	HPP
	✓

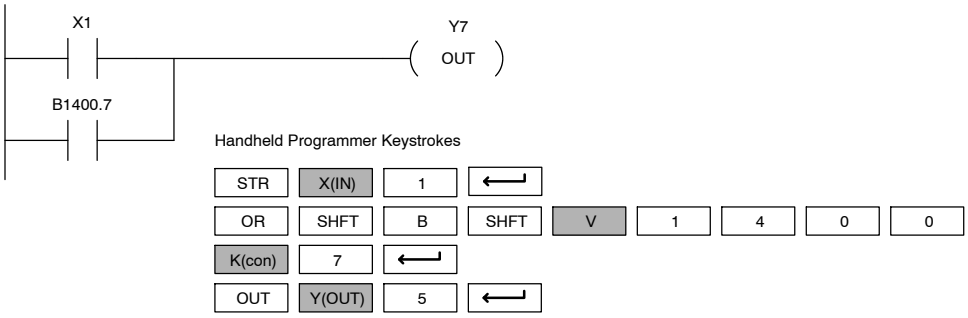
The Or Not Bit-of-Word instruction logically ors a normally closed contact in parallel with another contact in a rung. Status of the contact will be opposite the state of the bit referenced in the associated memory location.



Operand Data Type		DL450 Range	
	A	aaa	bb
V-memory	B	All (See p. 3-42)	BCD, 0 to 15
Pointer	PB	All (See p. 3-42)	BCD

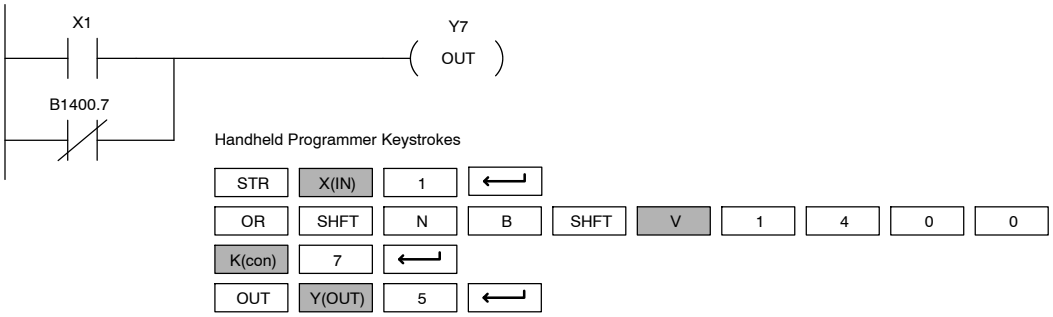
In the following Or Bit-of-Word example, when input X1 or bit 7 of V1400 is on, output Y5 will energize.

DirectSOFT



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

DirectSOFT

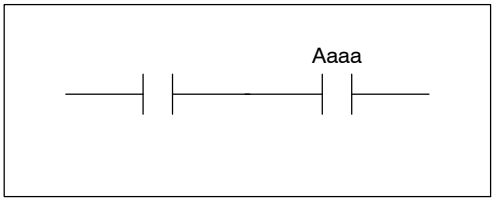


And (AND)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

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.

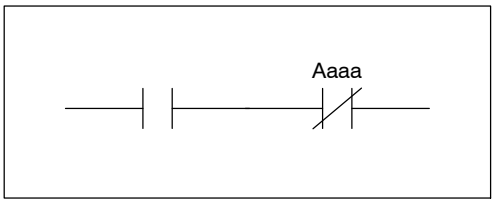


And Not (ANDN)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

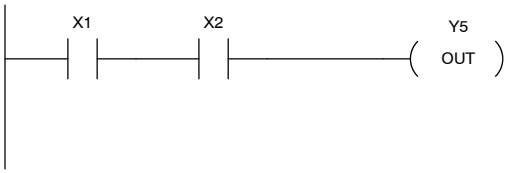
The And Not Bit-of-Word instruction logically ands a normally closed contact in series with another contact in a rung. The status of the contact will be opposite the state of the associated image register point or memory location.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
Inputs	X	0-477	0-477	0-1777
Outputs	Y	0-477	0-477	0-1777
Control Relays	C	0-737	0-1777	0-3777
Stage	S	0-577	0-1777	0-1777
Timer	T	0-177	0-377	0-377
Counter	CT	0-177	0-177	0-377
Special Relay	SP	0-137, 320-617	0-137 320-717	0-137, 320-717
Global	GX	0-777	0-1777	0-1777
Global	GY	-	-	0-1777

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

DirectSOFT

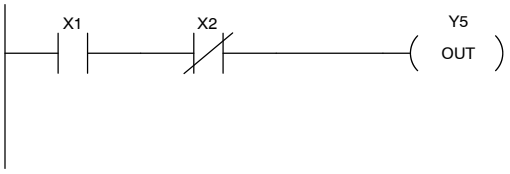


Handheld Programmer Keystrokes

STR	X(IN)	1	↵
AND	X(IN)	2	↵
OUT	Y(OUT)	5	↵

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

DirectSOFT



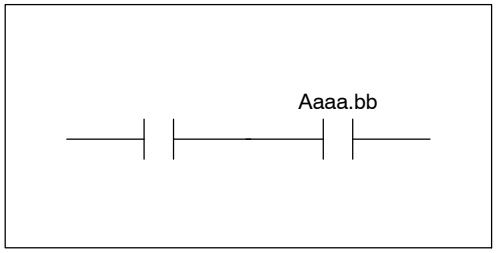
Handheld Programmer Keystrokes

STR	X(IN)	1	↵	
AND	NOT	X(IN)	2	↵
OUT	Y(OUT)	5	↵	

And Bit-of-Word (ANDB)

430	440	450
IMP		✓
DS		HPP

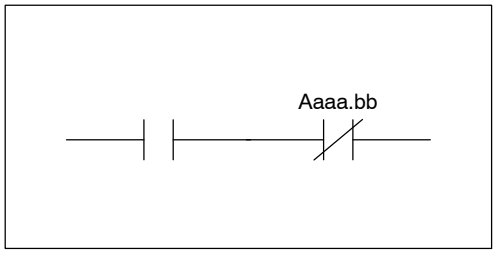
The And Bit-of-Word instruction logically ands a normally open contact in series with another contact in a rung. The status of the contact will be the same state as the bit referenced in the associated memory location.



And Not Bit-of-Word (ANDNB)

430	440	450
IMP		✓
DS		HPP

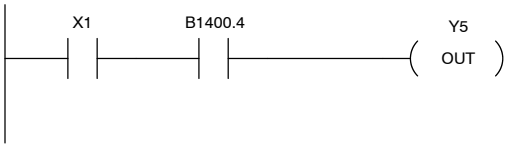
The And Not Bit-of-Word instruction logically ands a normally closed contact in series with another contact in a rung. The status of the contact will be opposite the state of the bit referenced in the associated memory location.



Operand Data Type		DL450 Range	
	A	aaa	bb
V-memory	B	All (See p. 3-42)	BCD, 0 to 15
Pointer	PB	All (See p. 3-42)	BCD

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

DirectSOFT

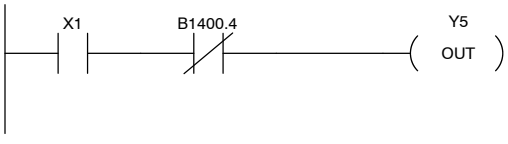


Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
AND	SHFT	B	SHFT	V	1	4	0	0	
K(con)	4	↵							
OUT	Y(OUT)	5	↵						

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.

DirectSOFT



Handheld Programmer Keystrokes

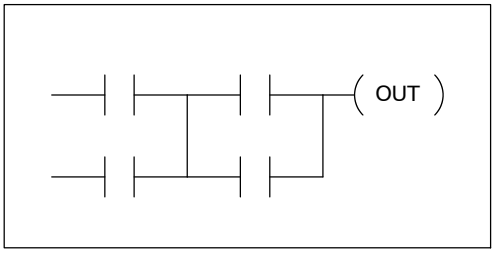
STR	X(IN)	1	↵						
AND	SHFT	N	B	SHFT	V	1	4	0	0
K(con)	4	↵							
OUT	Y(OUT)	5	↵						

And Store (ANDSTR)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

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

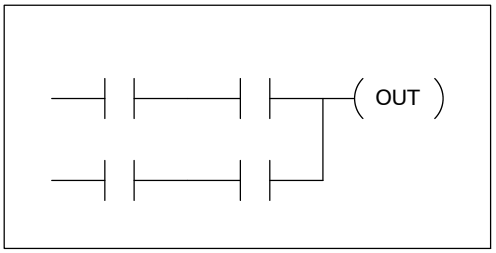


Or Store (ORSTR)

430	440	450
✓	✓	✓

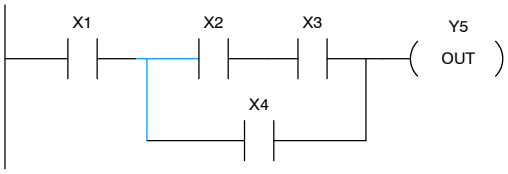
IMP	HPP
DS	✓

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.

DirectSOFT

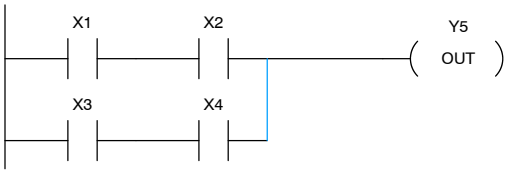


Handheld Programmer Keystrokes

STR	X(IN)	1	↵
STR	X(IN)	2	↵
AND	X(IN)	3	↵
OR	X(IN)	4	↵
AND	STR	↵	
OUT	Y(OUT)	5	↵

In the following Or Store example, the branch consisting of X1 and X2 have been ORed with the branch consisting of X3 and X4.

DirectSOFT



Handheld Programmer Keystrokes

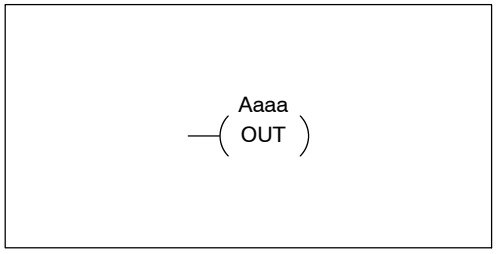
STR	X(IN)	1	↵
AND	X(IN)	2	↵
STR	X(IN)	3	↵
AND	X(IN)	4	↵
OR	STR	↵	
OUT	Y(OUT)	5	↵

Out (OUT)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

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.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	aaa	bb	bb
Inputs	X	0-477	0-477	0-1777
Outputs	Y	0-477	0-477	0-1777
Control Relays	C	0-737	0-1777	0-3777
Global I/O	GX	0-777	0-1777	0-2777 (GX + GY)

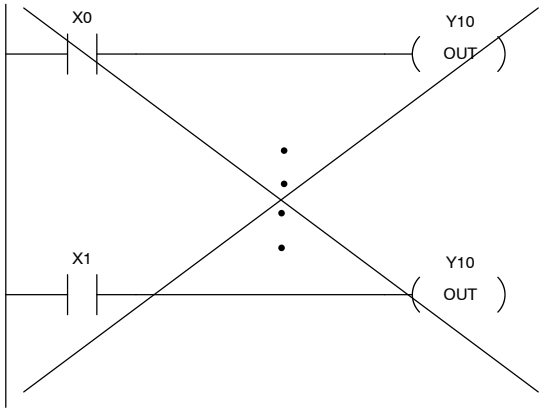
In the following Out example, when input X1 is on, output Y2 and Y5 will energize.

DirectSOFT

Handheld Programmer Keystrokes

STR	X(IN)	1	↵
OUT	Y(OUT)	2	↵
OUT	Y(OUT)	5	↵

In the following Out example the program contains two Out instructions using the same location (Y10). The physical output of Y10 is ultimately controlled by the last rung of logic referencing Y10. X1 will override the Y10 output being controlled by X0. To avoid this situation, multiple outputs using the same location should not be used in programming.

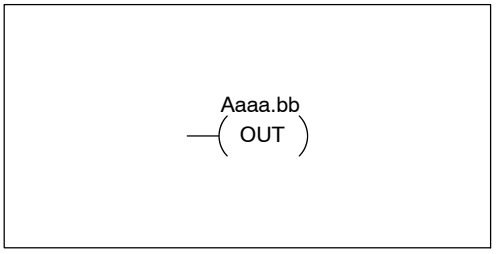


Out Bit-of-Word (OUTB)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

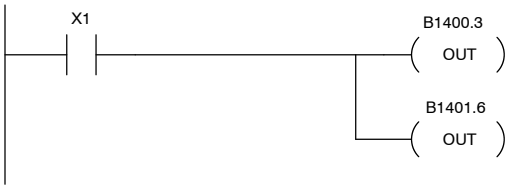
The Out Bit-of-Word instruction reflects the status of the rung (on/off) and outputs the discrete (on/off) state to the specified bit in the referenced memory location. Multiple Out Bit-of-Word instructions referencing the same bit of the same word generally should not be used since only the last Out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



Operand Data Type		DL450 Range	
	A	aaa	bb
V-memory	B	All (See p. 3-42)	BCD, 0 to 15
Pointer	PB	All (See p. 3-42)	BCD

In the following Out Bit-of-Word example, when input X1 is on, bit 3 of V1400 and bit 6 of V1401 will turn on.

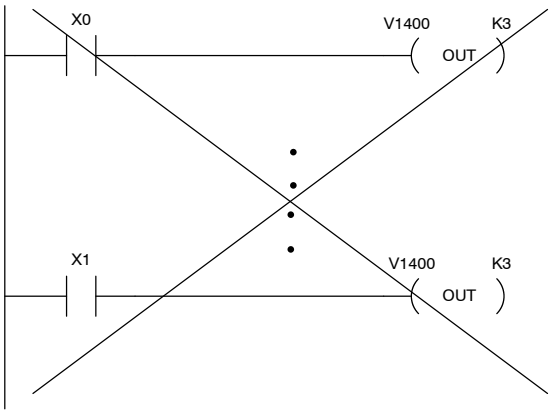
DirectSOFT



Handheld Programmer Keystrokes

STR	X(IN)	1	↵								
OUT	SHFT	B	SHFT	V	1	4	0	0	K(con)	3	↵
OUT	SHFT	B	SHFT	V	1	4	0	1	K(con)	6	↵

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

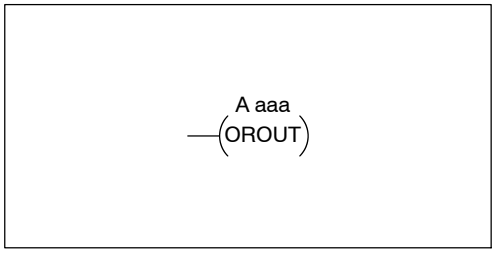


Or Out (OROUT)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

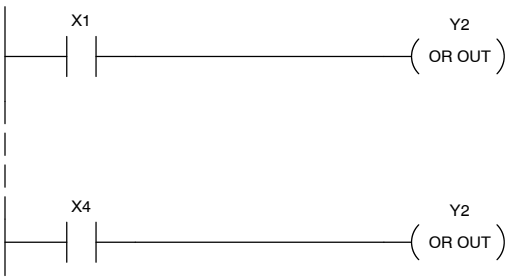
The Or Out instruction has been designed to use more than 1 rung of discrete logic to control a single output. Multiple Or Out instructions referencing the same output coil may be used, since *all* contacts controlling the output are ORed together. If the status of *any* rung is on, the output will also be on.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	aaa	aaa	aaa
Inputs	X	0-477	0-477	0-1777
Outputs	Y	0-477	0-477	0-1777
Control Relays	C	0-737	0-1777	0-3777
Global I/O	GX	0-777	0-1777	0-2777 (GX + GY)

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

DirectSOFT



Handheld Programmer Keystrokes

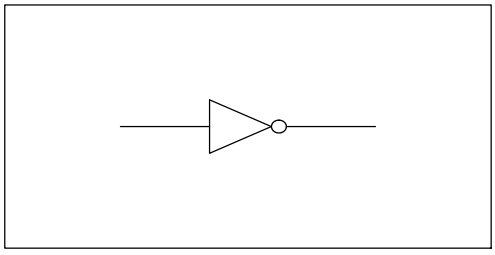
STR	X(IN)	1	↵
OR	OUT	Y(OUT)	2 ↵
STR	X(IN)	4	↵
OR	OUT	Y(OUT)	2 ↵

Not (NOT)

430	440	450
✓	✓	✓

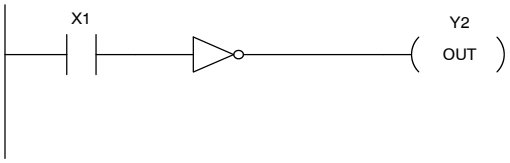
DS	HPP
✓	✓

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.

DirectSOFT



Handheld Programmer Keystrokes

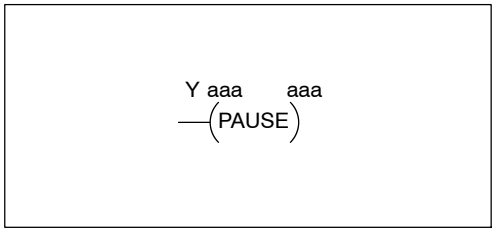
STR	X(IN)	1	↵
SHFT	N	O	T ↵
OUT	Y(OUT)	2	↵

NOTE: ***Direct***SOFT Release 1.1i and later supports the use of the NOT instruction. The above example rung is merely intended to show the visual representation of the NOT instruction. The rung cannot be created or displayed in ***Direct***SOFT versions earlier than 1.1i.

Pause (PAUSE)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

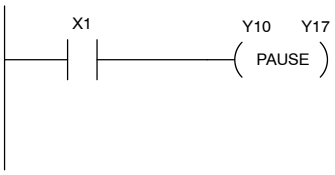
The Pause instruction disables the output update on a range of outputs. The ladder program will continue to run and update the image register however the outputs in the range specified in the Pause instruction will be turned off at the output module.



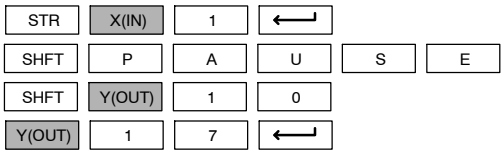
Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
Outputs	Y	0-477	0-477	0-1777

In the following example, when X1 is ON, Y10-Y17 will be turned OFF at the output module. The execution of the ladder program will not be affected.

DirectSOFT



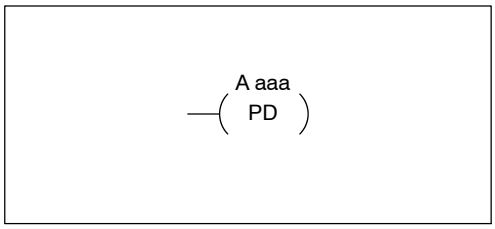
Handheld Programmer Keystrokes



Positive Differential (PD)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

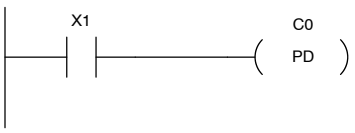
The Positive Differential instruction is typically known as a one-shot. When the input logic produces an Off-to-On transition, the output will energize for one CPU scan. Thereafter, it remains off until its input makes another Off-to-On transition.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
Inputs	X	0-477	0-477	0-1777
Outputs	Y	0-477	0-477	0-1777
Control Relays	C	0-737	0-1777	0-3777

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

DirectSOFT



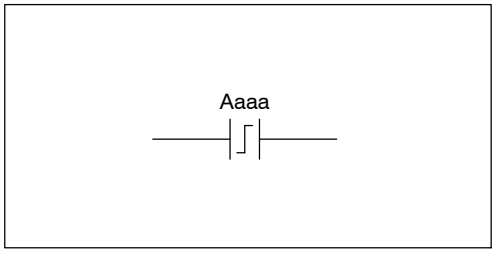
Handheld Programmer Keystrokes

Note that you can place a "NOT" instruction immediately before the PD instruction to generate a "one-shot" pulse on an on-to-off transition.

Store Positive Differential (STRPD)

430	440	450
✓		✓
DS		HPP

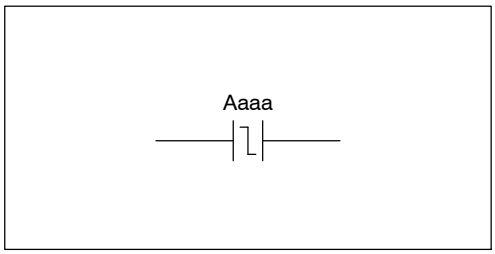
The Store Positive Differential instruction begins a new rung or an additional branch in a rung with a normally open contact. The contact closes for one CPU scan when the state of the associated image register point makes an Off-to-On transition. Thereafter, the contact remains open until the next Off-to-On 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 retentive range.



Store Negative Differential (STRND)

430	440	450
✓		✓
DS		HPP

The Store Negative Differential instruction begins a new rung or an additional branch in a rung with a normally closed 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		DL450 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
Global	GX	0-2777 (GX + GY)

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

DirectSOFT

Handheld Programmer Keystrokes

STR	SHFT	P	D	SHFT
X(IN)	1	↵		
OUT	Y(OUT)	4	↵	

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

DirectSOFT

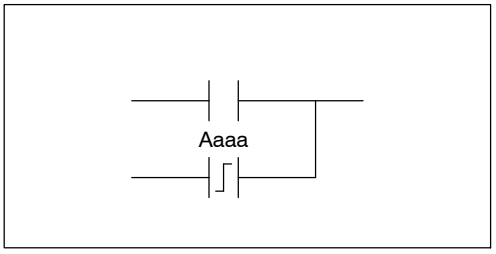
Handheld Programmer Keystrokes

STR	SHFT	N	D	SHFT
X(IN)	1	↵		
OUT	Y(OUT)	4	↵	

Or Positive Differential (ORPD)

430	440	450
IMP		✓
DS		HPP

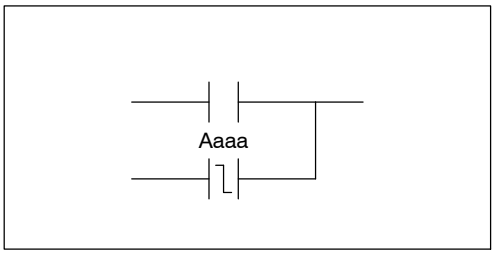
The Or Positive Differential instruction logically ors a normally open contact in parallel with another contact in a rung. The status of the contact will be open until the associated image register point makes an Off-to-On transition, closing it for one CPU scan. Thereafter, it remains open until another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



Or Negative Differential (ORND)

430	440	450
IMP		✓
DS		HPP

The Or Negative Differential instruction logically ors a normally open 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		DL450 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
Global	GX	0-2777 (GX + GY)

In the following example, Y 5 will energize whenever X1 is on, or for one CPU scan when X2 transitions from Off to On.

DirectSOFT

Handheld Programmer Keystrokes

STR	X(IN)	1	↵	
OR	SHFT	P	D	SHFT
X(IN)	2	↵		
OUT	Y(OUT)	5	↵	

In the following example, Y 5 will energize whenever X1 is on, or for one CPU scan when X2 transitions from On to Off.

DirectSOFT

Handheld Programmer Keystrokes

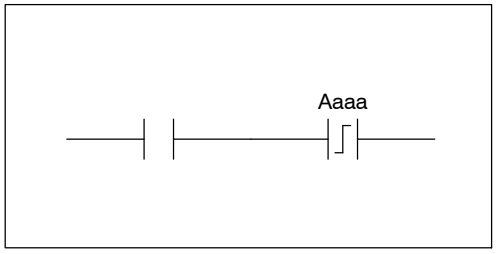
STR	X(IN)	1	↵	
OR	SHFT	N	D	SHFT
X(IN)	2	↵		
OUT	Y(OUT)	5	↵	

And Positive Differential (ANDPD)

430	440	450
✗	✗	✓

IMP	✓
DS	HPP

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.

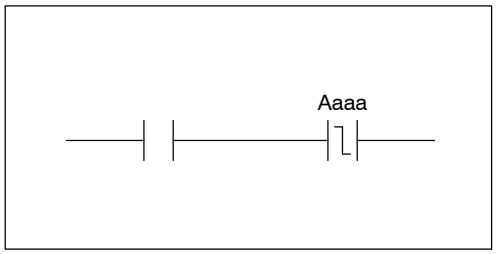


And Negative Differential (ANDND)

430	440	450
✗	✗	✓

IMP	✓
DS	HPP

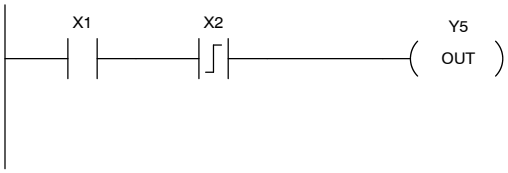
The And Negative Differential instruction logically ands a normally open contact in series with another contact in a rung. The status of the contact will be open until the associated image register point makes an On-to-Off transition, closing it for one CPU scan. Thereafter, it remains open until another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



Operand Data Type		DL450 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
Global	GX	0-2777 (GX + GY)

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

DirectSOFT

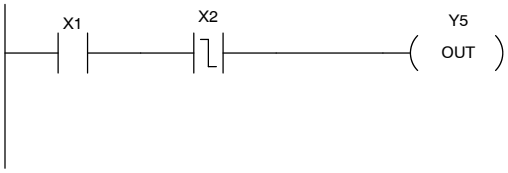


Handheld Programmer Keystrokes

STR	X(IN)	1	↵	
AND	SHFT	P	D	SHFT
X(IN)	2	↵		
OUT	Y(OUT)	5	↵	

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

DirectSOFT



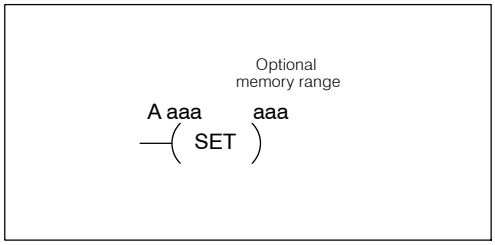
Handheld Programmer Keystrokes

STR	X(IN)	1	↵	
AND	SHFT	N	D	SHFT
X(IN)	2	↵		
OUT	Y(OUT)	5	↵	

Set (SET)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

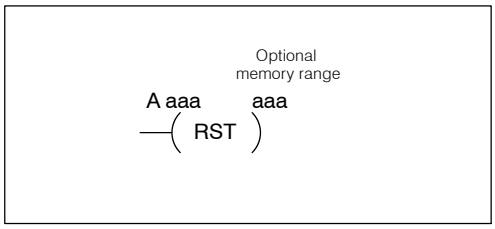
The Set instruction sets or turns on an image register point/memory location or a consecutive range of image register points/memory locations. Once the point/location is set it will remain on until it is reset using the Reset instruction. It is not necessary for the input controlling the Set instruction to remain on.



Reset (RST)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

The Reset instruction resets or turns off an image register point/memory location or a range of image registers points/memory locations. Once the point/location is reset it is not necessary for the input to remain on.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
Inputs	X	0-477	0-477	0-1777
Outputs	Y	0-477	0-477	0-1777
Control Relays	C	0-737	0-1777	0-3777
Stages	S	0-577	0-1777	0-1777
Timers*	T	0-177	0-377	0-377
Counters*	CT	0-177	0-177	0-377
Global	GX	0-777	0-1777	0-2777 (GX + GY)

* Timer and counter operand data types are not valid using the Set instruction.

In the following example when X1 turns on, Y5 through Y22 will be set to the on state.

DirectSOFT

Handheld Programmer Keystrokes

STR	X(IN)	1	↵			
SET	Y(OUT)	5	Y(OUT)	2	2	↵

In the following example when X1 turns on, Y5 through Y22 will be reset to the off state.

DirectSOFT

Handheld Programmer Keystrokes

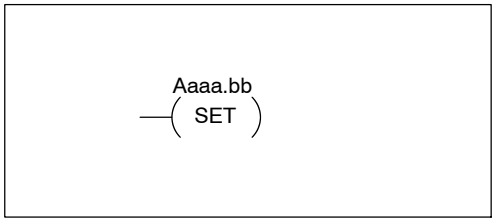
STR	X(IN)	1	↵			
RST	Y(OUT)	5	Y(OUT)	2	2	↵

Set Bit-of-Word (SETB)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

The Set Bit-of-Word instruction sets or turns on a bit in a V-memory location. Once the bit is set it will remain on until it is reset using the Reset Bit-of-Word instruction. It is not necessary for the input controlling the Set Bit-of-Word instruction to remain on.

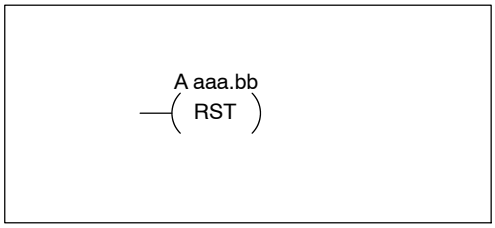


Reset Bit-of-Word (RSTB)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

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		DL450 Range	
	A	aaa	bb
V-memory	B	All (See p. 3-42)	0 to 15
Pointer	PB	All (See p. 3-42)	0 to 15

In the following example when X1 turns on, bit 0 in V1400 is set to the on state.

DirectSOFT

Handheld Programmer Keystrokes

STR	X(IN)	1	↵					
SET	SHFT	B	SHFT	V	1	4	0	0
K(con)	0	↵						

In the following example when X1 turns on, bit 15 in V1400 is reset to the off state.

DirectSOFT

Handheld Programmer Keystrokes

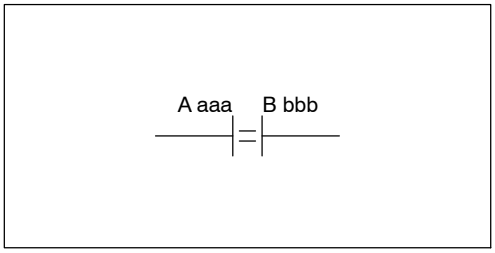
STR	X(IN)	1	↵					
RST	SHFT	B	SHFT	V	1	4	0	0
K(con)	1	5	↵					

Comparative Boolean Instructions

Store If Equal (STRE)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

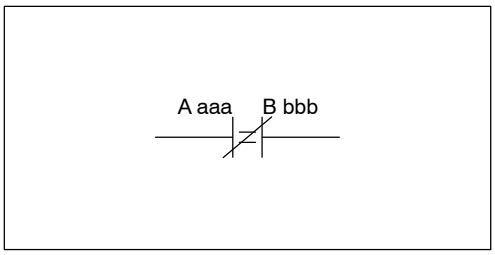
The Store If Equal instruction begins a new rung or additional branch in a rung with a normally open comparative contact. The contact will be on when Aaaa = Bbbb.



Store If Not Equal (STRNE)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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



Operand Data Type		DL430 Range		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All (See p. 3-40)	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Pointer	P	--	--	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Constant	K	--	0-FFFF	--	0-FFFF	--	0-FFFF

In the following example, when the value in V-memory location V1400 = 4933 , Y3 will energize.

DirectSOFT

Handheld Programmer Keystrokes

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	4	9	3	3	↵			
OUT	Y(OUT)	3	↵					

In the following example, when the value in V-memory location V1400 ≠ 5060, Y3 will energize.

DirectSOFT

Handheld Programmer Keystrokes

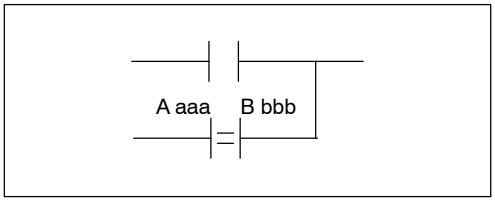
STR	NOT	SHFT	E	SHFT	V	1	4	0	0
K(CON)	5	0	6	0	↵				
OUT	Y(OUT)	3	↵						

Or If Equal (ORE)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

The Or If Equal instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa = Bbbb.

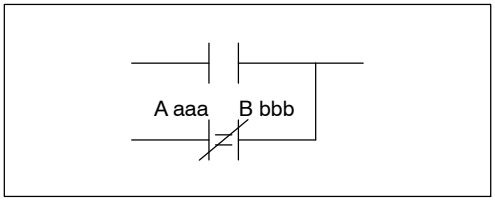


Or If Not Equal (ORNE)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

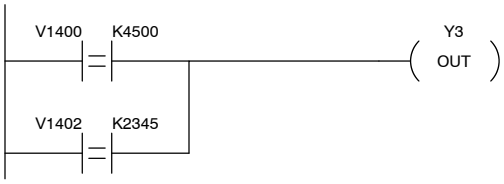
The Or If Not Equal instruction connects a normally closed comparative contact in parallel with another contact. The contact will be on when Aaaa ≠ Bbbb.



Operand Data Type		DL430 Range		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All (See p. 3-40)	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Pointer	P	--	--	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Constant	K	--	0-FFFF	--	0-FFFF	--	0-FFFF

In the following example, when the value in V-memory location V1400 = 4500 or V1402 = 2345 , Y3 will energize.

DirectSOFT

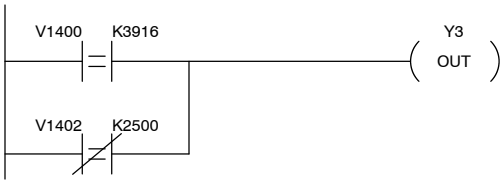


Handheld Programmer Keystrokes

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	4	5	0	0	↵			
OR	SHFT	E	SHFT	V	1	4	0	2
K(CON)	2	3	4	5	↵			
OUT	Y(OUT)	3	↵					

In the following example, when the value in V-memory location V1400 = 3916 or V1402 ≠ 2500, Y3 will energize.

DirectSOFT



Handheld Programmer Keystrokes

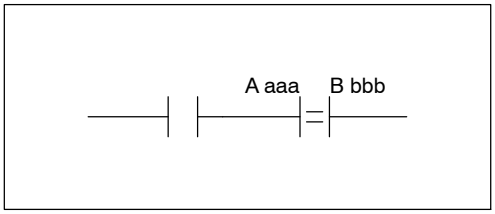
STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	3	9	1	6	↵			
OR	NOT	SHFT	E	SHFT	V	1	4	0	2
K(CON)	2	5	0	0	↵			
OUT	Y(OUT)	3	↵					

And If Equal (ANDE)

430	440	450
✓	✓	✓

IMP	HPP
	✓
DS	

The And If Equal instruction connects a normally open comparative contact in series with another contact. The contact will be on when Aaaa = Bbbb.

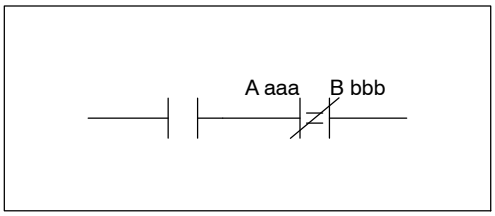


And If Not Equal (ANDNE)

430	440	450
✓	✓	✓

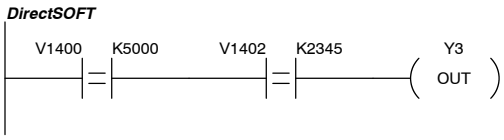
IMP	HPP
	✓
DS	

The And If Not Equal instruction connects a normally closed comparative contact in series with another contact. The contact will be on when Aaaa ≠ Bbbb



Operand Data Type		DL430 Range		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All (See p. 3-40)	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Pointer	P	--	--	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Constant	K	--	0-FFFF	--	0-FFFF	--	0-FFFF

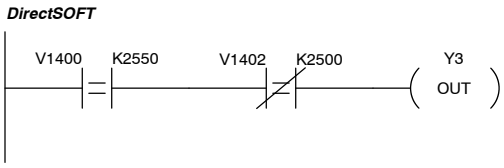
In the following example, when the value in V-memory location V1400 = 5000 and V1402 = 2345, Y3 will energize.



Handheld Programmer Keystrokes

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	5	0	0	0	↵			
AND	SHFT	E	SHFT	V	1	4	0	2
K(CON)	2	3	4	5	↵			
OUT	Y(OUT)	3	↵					

In the following example, when the value in V-memory location V1400 = 2550 and V1402 ≠ 2500, Y3 will energize.



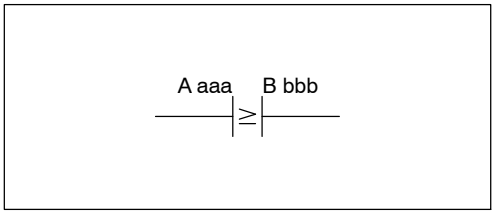
Handheld Programmer Keystrokes

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	2	5	5	0	↵			
AND	NOT	SHFT	E	SHFT	V	1	4	0	2
K(CON)	2	5	0	0	↵			
OUT	Y(OUT)	3	↵					

Store (STR)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

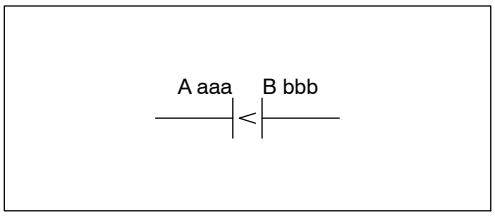
The Comparative Store instruction begins a new rung or additional branch in a rung with a normally open comparative contact. The contact will be on when Aaaa ≥ Bbbb.



Store Not (STRN)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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 < Bbbb.



Operand Data Type		DL430 Range		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All (See p. 3-40)	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Pointer	P	--	--	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Constant	K	--	0-FFFF	--	0-FFFF	--	0-FFFF
Timer	T	0-177		0-377		0-377	
Counter	CT	0-177		0-177		0-377	

In the following example, when the value in V-memory location V1400 ≥ 1000, Y3 will energize.

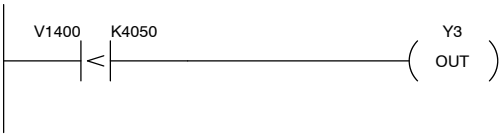
DirectSOFT

Handheld Programmer Keystrokes

STR	V	1	4	0	0
K(CON)	1	0	0	0	↵
OUT	Y(OUT)	3	↵		

In the following example, when the value in V-memory location V1400 < 4050, Y3 will energize.

DirectSOFT



Handheld Programmer Keystrokes

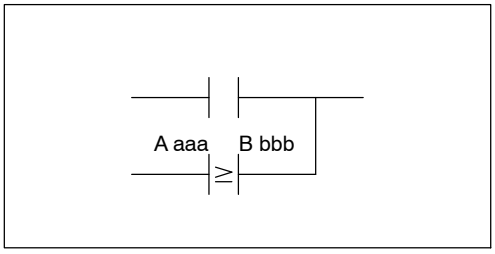
STR	NOT	V	1	4	0	0
K(CON)	4	0	5	0	↵	
OUT	Y(OUT)	3	↵			

Or (OR)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

The Comparative Or instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa ≥ Bbbb.

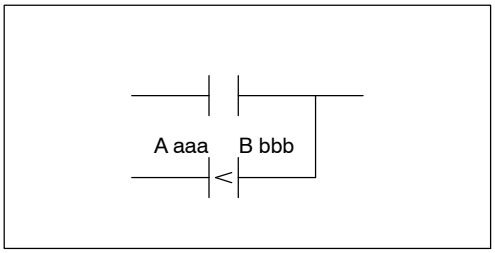


Or Not (ORN)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

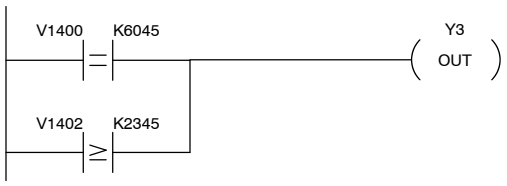
The Comparative Or Not instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa < Bbbb.



Operand Data Type		DL430 Range		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All (See p. 3-40)	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Pointer	P	--	--	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Constant	K	--	0-FFFF	--	0-FFFF	--	0-FFFF
Timer	T	0-177		0-377		0-377	
Counter	CT	0-177		0-177		0-377	

In the following example, when the value in V-memory location V1400 = 6045 or V1402 ≥ 2345, Y3 will energize.

DirectSOFT

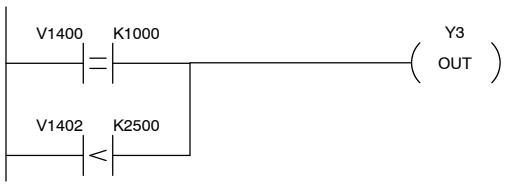


Handheld Programmer Keystrokes

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	6	0	4	5	↵			
OR	V	1	4	0	2			
K(CON)	2	3	4	5	↵			
OUT	Y(OUT)	3	↵					

In the following example when the value in V-memory location V1400 = 1000 or V1402 < 2500, Y3 will energize.

DirectSOFT



Handheld Programmer Keystrokes

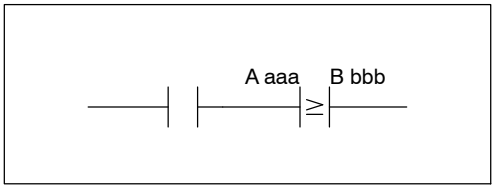
STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	1	0	0	0	↵			
OR	NOT	V	1	4	0	2		
K(CON)	2	5	0	0	↵			
OUT	Y(OUT)	3	↵					

And (AND)

✓ 430 ✓ 440 ✓ 450

IMP DS ✓ HPP

The Comparative And instruction connects a normally open comparative contact in series with another contact. The contact will be on when Aaaa ≥ Bbbb.

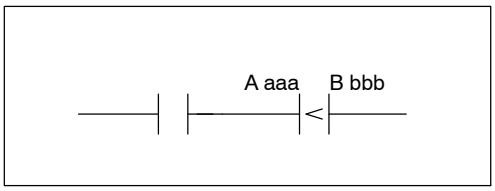


And Not (ANDN)

✓ 430 ✓ 440 ✓ 450

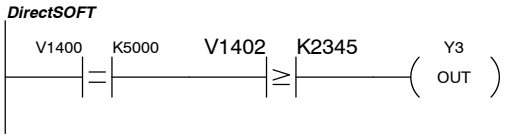
IMP DS ✓ HPP

The Comparative And Not instruction connects a normally open comparative contact in series with another contact. The contact will be on when Aaaa < Bbbb.



Operand Data Type		DL430 Range		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb	aaa	bbb
V-memory	V	All (See p. 3-40)	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Pointer	P	--	--	All (See p. 3-41)	All (See p. 3-41)	All (See p. 3-42)	All (See p. 3-42)
Constant	K	--	0-FFFF	--	0-FFFF	--	0-FFFF
Timer	T	0-177		0-377		0-377	
Counter	CT	0-177		0-177		0-377	

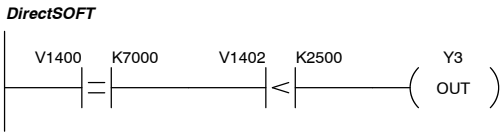
In the following example, when the value in V-memory location V1400 = 5000, and V1402 ≥ 2345, Y3 will energize.



Handheld Programmer Keystrokes

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	5	0	0	0	↵			
AND	V	1	4	0	2			
K(CON)	2	3	4	5	↵			
OUT	Y(OUT)	3	↵					

In the following example, when the value in V-memory location V1400 = 7000 and V1402 < 2500, Y3 will energize.



Handheld Programmer Keystrokes

STR	SHFT	E	SHFT	V	1	4	0	0
K(CON)	7	0	0	0	↵			
AND	NOT	V	1	4	0	2		
K(CON)	2	5	0	0	↵			
OUT	Y(OUT)	3	↵					

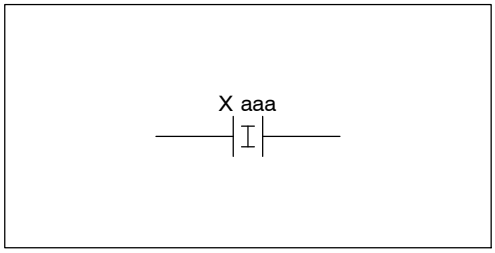
Immediate Instructions

Store Immediate (STRI)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

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 on the module *at the time the instruction is executed*. The image register is not updated.

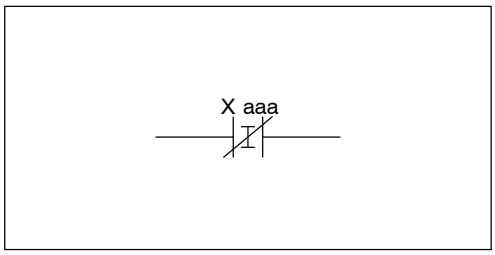


Store Not Immediate (STRNI)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

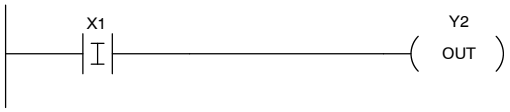
The Store Not Immediate instruction begins a new rung or additional branch in a rung. The status of the contact will be opposite the status of the associated input point on the module *at the time the instruction is executed*. The image register is not updated.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Inputs	X	0-477	0-477	0-1777

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

DirectSOFT



Handheld Programmer Keystrokes

STR	SHFT	I	SHFT	X(IN)	1	↵
OUT	Y(OUT)	2	↵			

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

DirectSOFT

Handheld Programmer Keystrokes

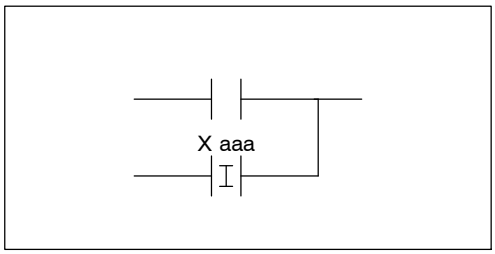
STR	NOT	SHFT	I	SHFT	X(IN)	1	↵
OUT	Y(OUT)	2	↵				

Or Immediate (ORI)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

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 on the module *at the time the instruction is executed*. The image register is not updated.

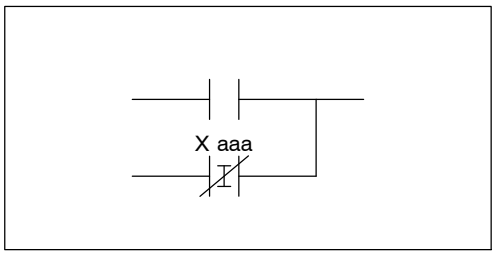


Or Not Immediate (ORNI)

430	440	450
✓	✓	✓

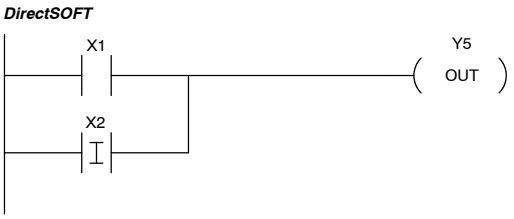
IMP	HPP
DS	✓

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



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Inputs	X	0-477	0-477	0-1777

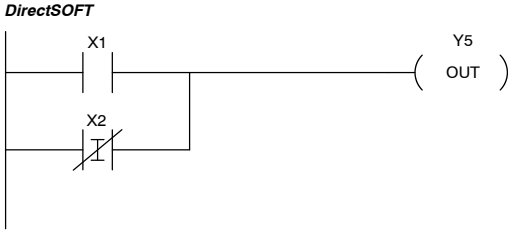
In the following example, when X1 or X2 is on, Y5 will energize.



Handheld Programmer Keystrokes

STR	X(IN)	1	↵				
OR	SHFT	I	SHFT	X(IN)	2	↵	
OUT	Y(OUT)	5	↵				

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



Handheld Programmer Keystrokes

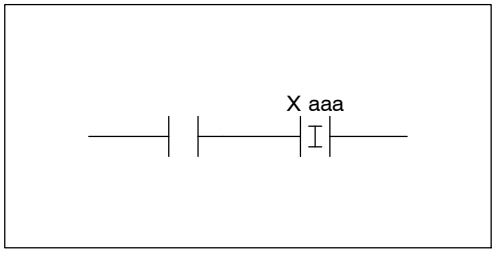
STR	X(IN)	1	↵				
OR	NOT	SHFT	I	SHFT	X(IN)	2	↵
OUT	Y(OUT)	5	↵				

And Immediate (ANDI)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

The And Immediate connects two contacts in series. The status of the contact will be the same as the status of the associated input point on the module *at the time the instruction is executed*. The image register is not updated.

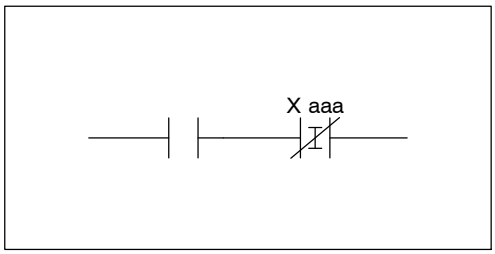


And Not Immediate (ANDNI)

430	440	450
✓	✓	✓

IMP	HPP
DS	✓

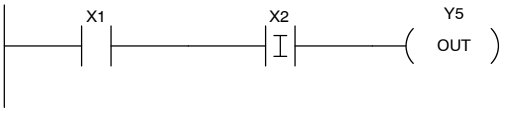
The And Not Immediate connects two contacts in series. The status of the contact will be opposite the status of the associated input point on the module *at the time the instruction is executed*. The image register is not updated.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Inputs	X	0-477	0-477	0-1777

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

DirectSOFT

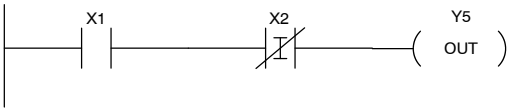


Handheld Programmer Keystrokes

STR	X(IN)	1	↵				
AND	SHFT	I	SHFT	X(IN)	2	↵	
OUT	Y(OUT)	5	↵				

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

DirectSOFT



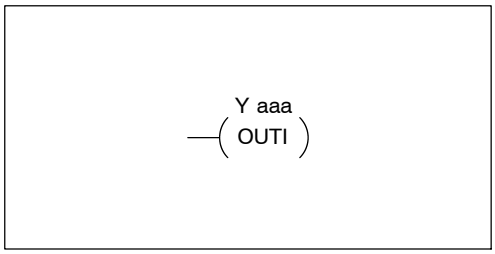
Handheld Programmer Keystrokes

STR	X(IN)	1	↵				
AND	NOT	SHFT	I	SHFT	X(IN)	2	↵
OUT	Y(OUT)	5	↵				

Out Immediate (OUTI)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

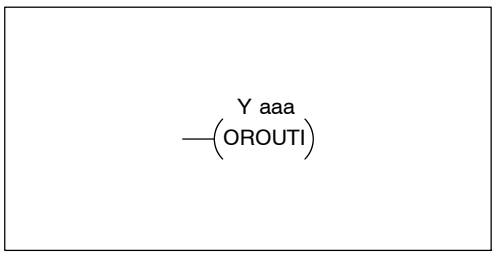
The Out Immediate instruction reflects the status of the rung (on/off) and outputs the discrete (on/off) status to the specified module output point and the image register *at the time the instruction is executed*. If multiple Out Immediate instructions referencing the same discrete point are used it is possible for the module output status to change multiple times in a CPU scan. See Or Out Immediate.



Or Out Immediate (OROUTI)

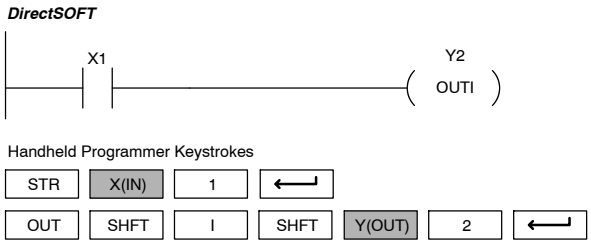
✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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 ored together. If the status of *any* rung is on *at the time the instruction is executed*, the output will also be on.

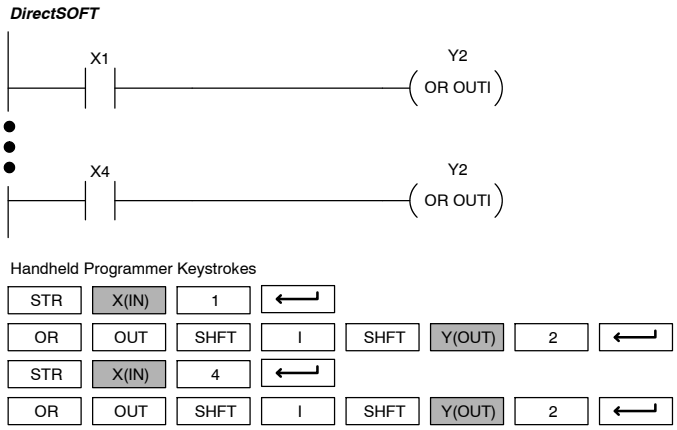


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Outputs	Y	0-477	0-477	0-1777

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



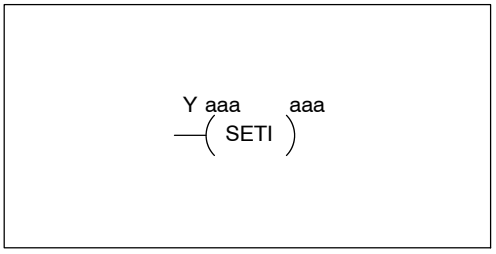
In the following example, when X1 or X4 is on, Y2 will energize.



Set Immediate (SETI)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

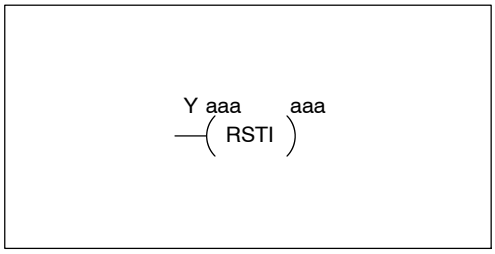
The Set Immediate instruction immediately sets or turns on an output or a range of outputs and the corresponding output module(s) *at the time the instruction is executed*. The image register is not updated. Once the outputs are set it is not necessary for the input to remain on. The Reset Immediate instruction can be used to reset the outputs.



Reset Immediate (RSTI)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

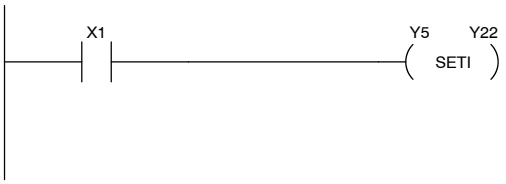
The Reset Immediate instruction immediately resets, or turns off an output or a range of outputs and the output module(s) *at the time the instruction is executed*. The image register is not updated. Once the outputs are reset it is not necessary for the input to remain on.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Outputs	Y	0-477	0-477	0-1777

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

DirectSOFT

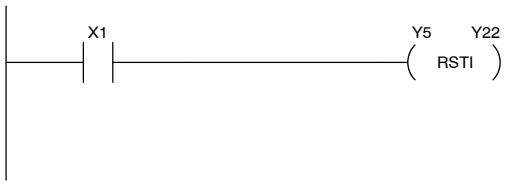


Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
SET	SHFT	I	SHFT	Y(OUT)	5	Y(OUT)	2	2	↵

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

DirectSOFT



Handheld Programmer Keystrokes

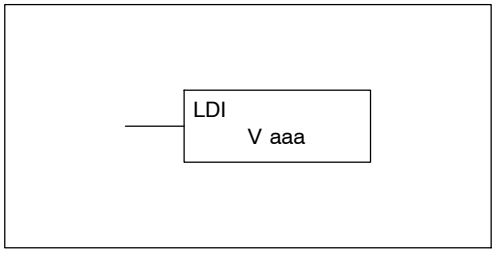
STR	X(IN)	1	↵						
RST	SHFT	I	SHFT	Y(OUT)	5	Y(OUT)	2	2	↵

Load Immediate (LDI)

430	440	450
✓		✓

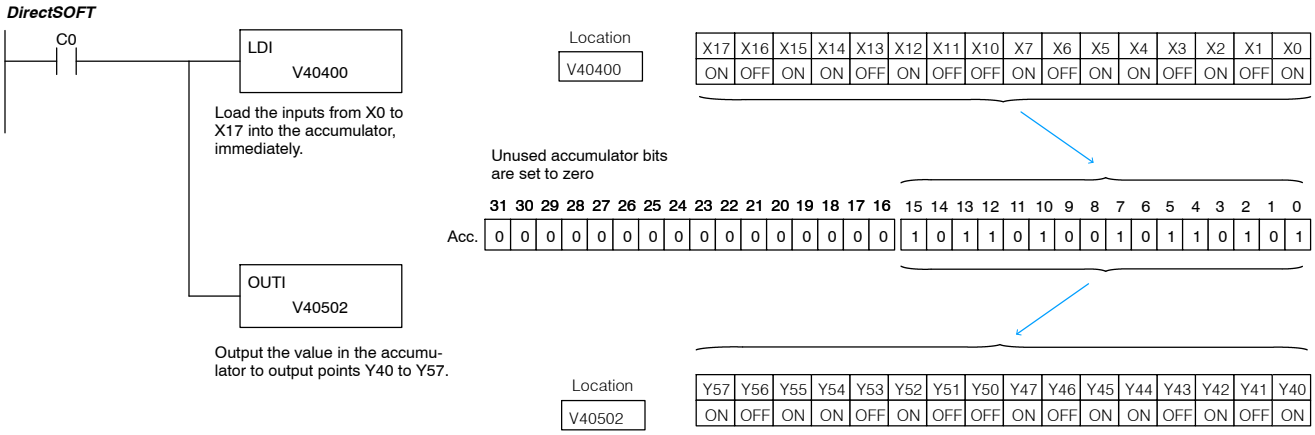
DS	HPP
✓	✓

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		DL450 Range
		aaaaa
Inputs	V	40400 - 40477

In the following example, when C0 is on, the binary pattern of X10-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 on a full CPU scan to occur).



Handheld Programmer Keystrokes

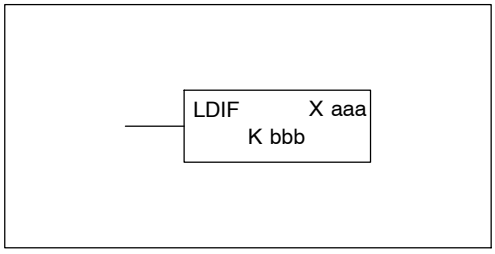
STR	C(CR)	0	↵							
LD	SHFT	I	SHFT	V	4	0	4	0	0	↵
OUT	SHFT	I	SHFT	V	4	0	5	4	0	↵

Load Immediate Formatted (LDIF)

430	440	450
	✓	✓

DS	HPP
✓	✓

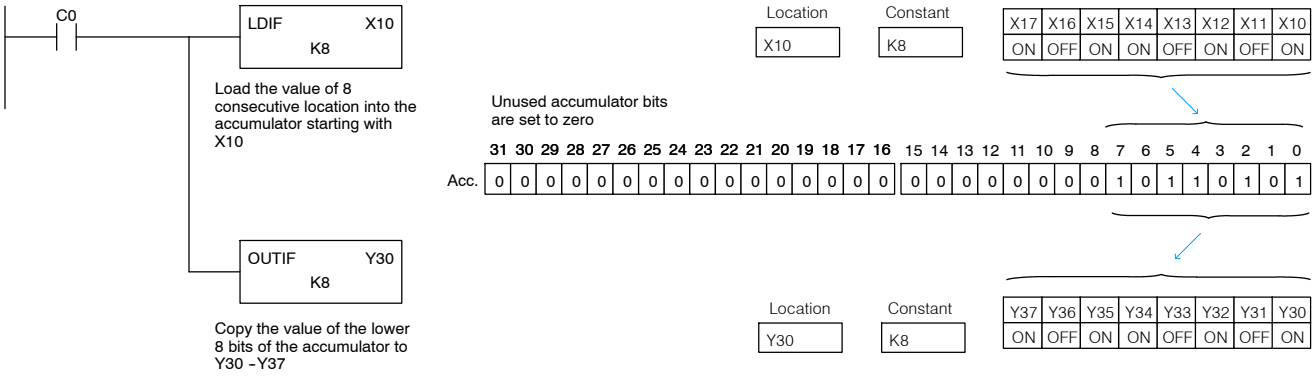
The Load Immediate Formatted instruction loads a 1-32 bit binary value into the accumulator. The value reflects the current status of the input module(s) *at the time the instruction is executed*. Accumulator bits that are not used by the instruction are set to zero.



Operand Data Type		DL440 Range		DL450 Range	
		aaa	bbb	aaa	bbb
Inputs	X	0-477	--	0-1777	--
Constant	K	--	1-32	--	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).

DirectSOFT



Handheld Programmer Keystrokes

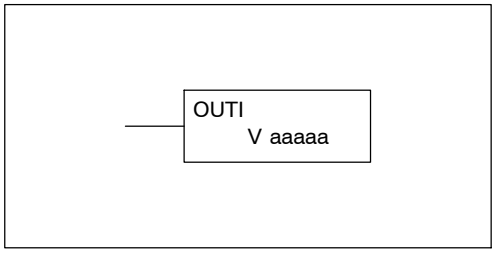
STR	C(CR)	0	↵							
LD	SHFT	I	F	SHFT	X(IN)	1	0	K(CON)	8	↵
OUT	SHFT	I	F	SHFT	Y(OUT)	3	0	K(CON)	8	↵

Out Immediate (OUTI)

430	440	450
✓		✓

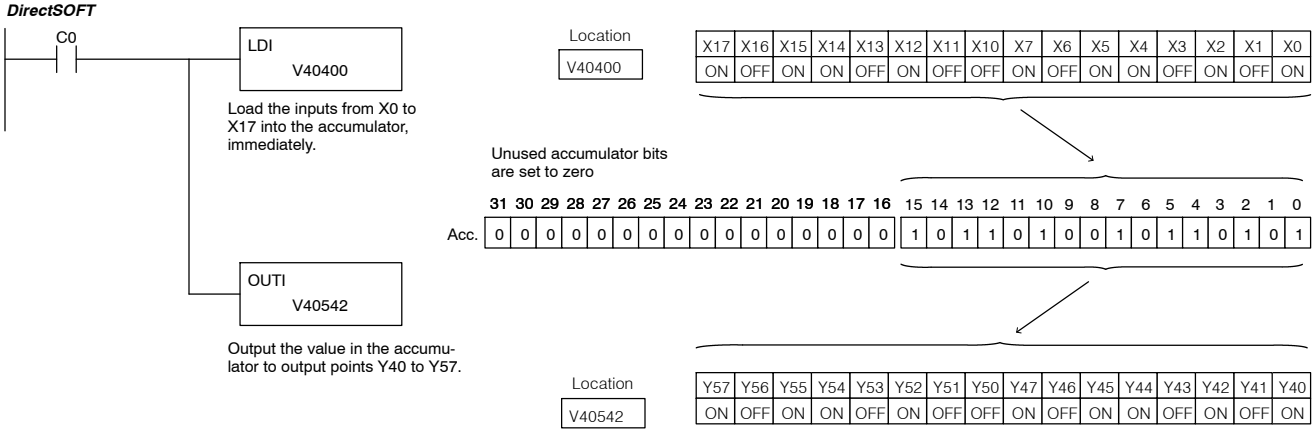
DS HPP

The Out Immediate instruction outputs a 16-bit binary value from the accumulator to a V-memory address *at the time the instruction is executed*. The valid address range includes all output point addresses on the local base. This instruction may be used instead of the OUTIF instruction which requires you to specify the number of input points.



Operand Data Type		DL450 Range
		aaaaa
Inputs	V	40400 - 40477

In the following example, when C0 is on, the binary pattern of X10–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 outputs (without waiting for the CPU scan).



Handheld Programmer Keystrokes

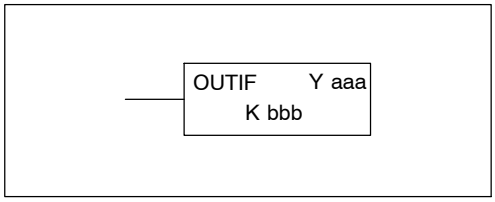
STR	C(CR)	0	↵							
LD	SHFT	I	SHFT	V	4	0	4	0	0	↵
OUT	SHFT	I	SHFT	V	4	0	5	4	0	↵

Out Immediate Formatted (OUTIF)

430	440	450
✗	✓	✓

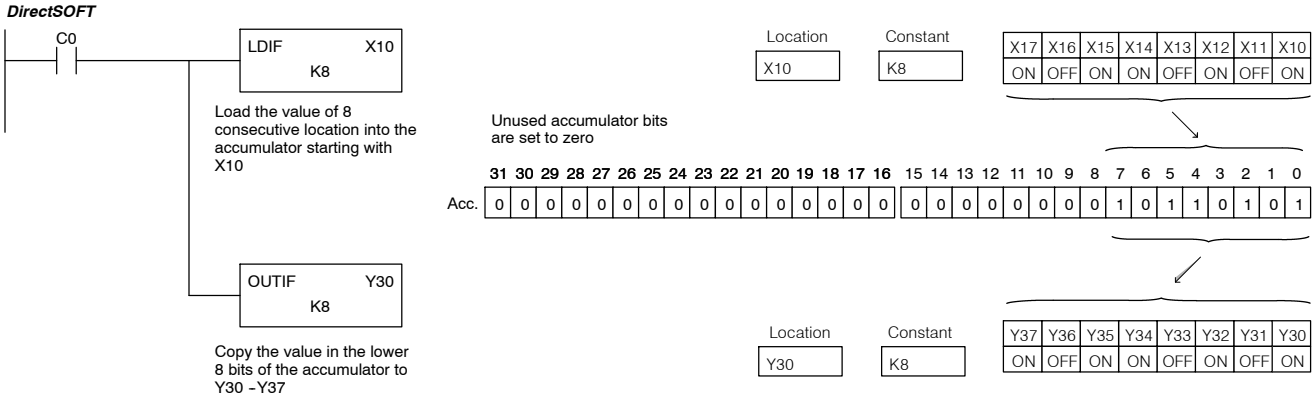
DS	HPP
✓	✓

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		DL440 Range		DL450 Range	
		aaa	bbb	aaa	bbb
Output	Y	0-477	--	0-1777	--
Constant	K	--	1-32	--	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).



Handheld Programmer Keystrokes

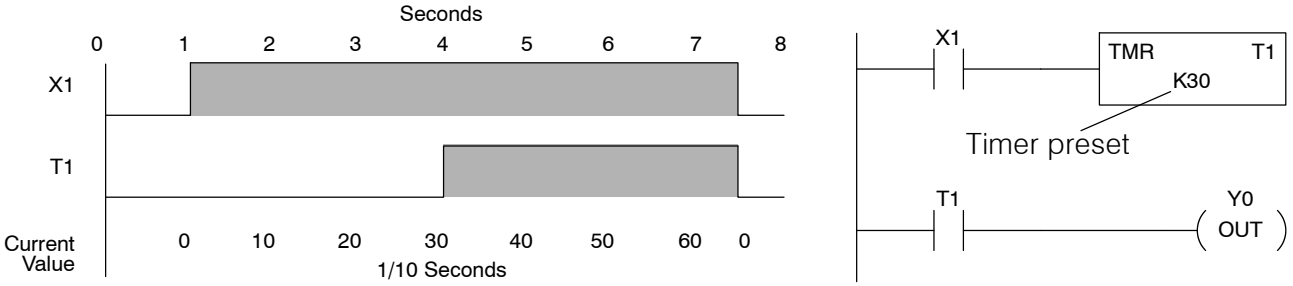
STR	C(CR)	0	↵							
LD	SHFT	I	F	SHFT	X(IN)	1	0	K(CON)	8	↵
OUT	SHFT	I	F	SHFT	Y(OUT)	3	0	K(CON)	8	↵

Timer, Counter, and Shift Register Instructions

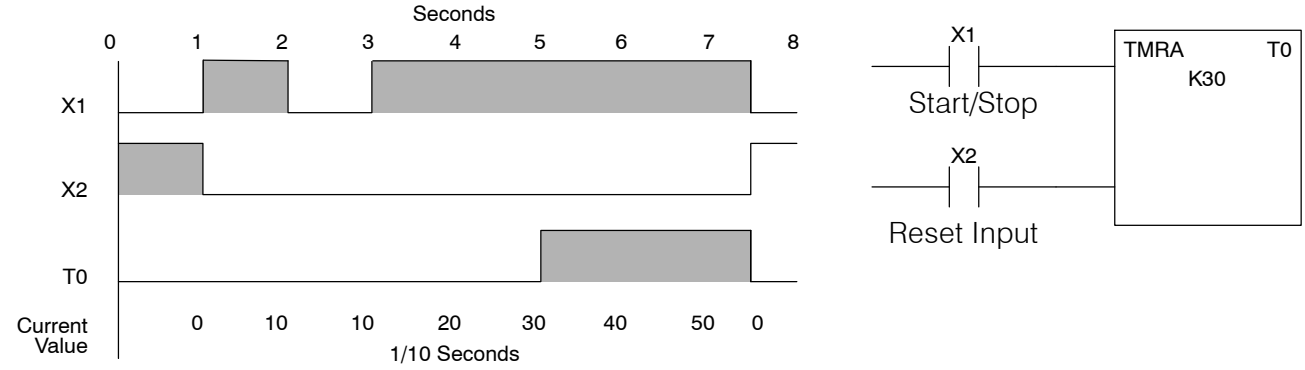
Using Timers

Timers are used to time an event for a desired length of time. 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 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 discrete bit associated with each timer to indicate 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.



The accumulating timer works similarly to the regular timer, but two inputs are required. The start/stop 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.

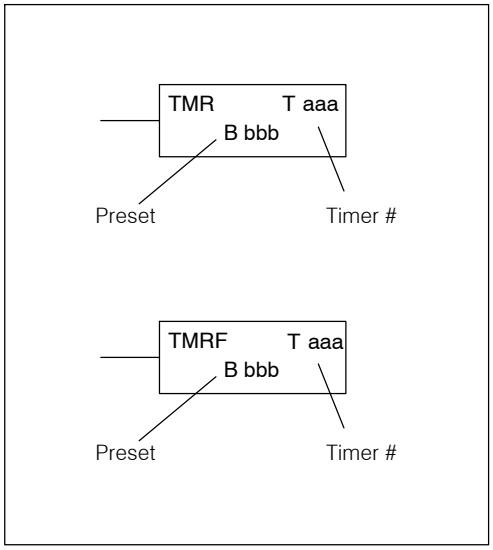


Timer (TMR) and Timer Fast (TMRF)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

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).



The timer discrete status bit and the current value is not specified in the timer instruction.

Instruction Specifications

Timer Reference (Taaa): Specifies the timer number.

Preset Value (Bbbb): Constant value (K) or a V-memory location. (V locations are 16-bit words.)

Current Value: Timer current values are accessed by referencing the associated V or T memory location*. For example, the timer current value for T3 physically resides in V-memory location V3. (V locations are 16-bit words.)

Discrete Status Bit: The discrete status bit is accessed by referencing the associated T memory location. It will be on if the current value is equal to or greater than the preset value. For example the discrete status bit for timer 2 would be T2.

Operand Data Type		DL430 Range		DL440 Range		DL450 Range	
	B	aaa	bbb	aaa	bbb	aaa	bbb
Timers	T	0-177	--	0-377	--	0-377	--
V-memory for preset values	V	--	1400-7377	--	1400-7377 10000-17777	--	1400-7377 10000-17777
Pointers (preset only)	P	--	--	--	1400-7377 10000-17777	--	1400-7377 10000-37777
Constants (preset only)	K	--	0-9999	--	0-9999	--	0-9999
Timer discrete status bits	T	0-177		0-377		0-377	
Timer current values	V /T*	0-177		0-377		0-377	

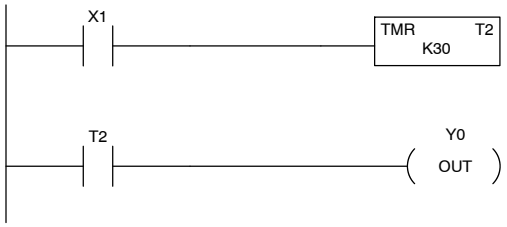
There are two methods of programming timers. You can perform functions when the timer reaches the specified preset using the discrete status bit, or use the comparative contacts to perform functions at different time intervals based on one timer. The following examples show each method of using timers.

NOTE: * For the Handheld Programmer, both the Timer discrete status bits and current value can be accessed with the same data type (example T2). The way the data type is used determines if it is a status bit or a current value. Any comparative instruction using T2 will access the current value, all other instructions using T2 will access the status bit. Current values may also be accessed by the V-memory location. For ***Direct***SOFT, the use of T2 will refer to the timer's discrete status bit. You should use V2 (or the alias TA2) to refer to the current value.

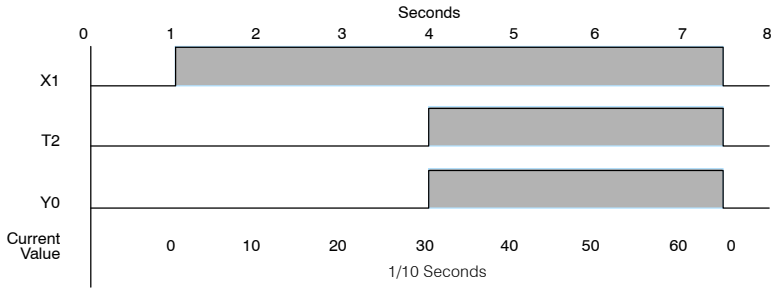
Timer Example Using Discrete Status Bits

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 after 7.5 seconds turning the discrete status bit off and resetting the timer current value to 0.

DirectSOFT



Timing Diagram



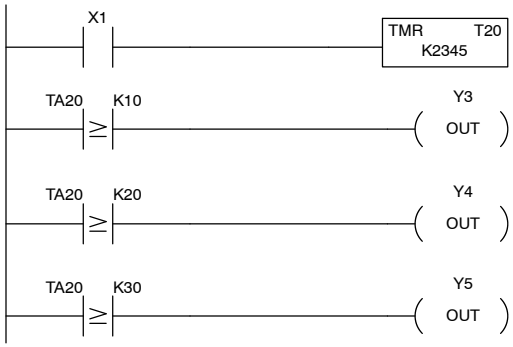
Handheld Programmer Keystrokes

STR	X(IN)	1	↵			
TMR	TMR	2	K(CON)	3	0	↵
STR	TMR	2	↵			
OUT	Y(OUT)	0	↵			

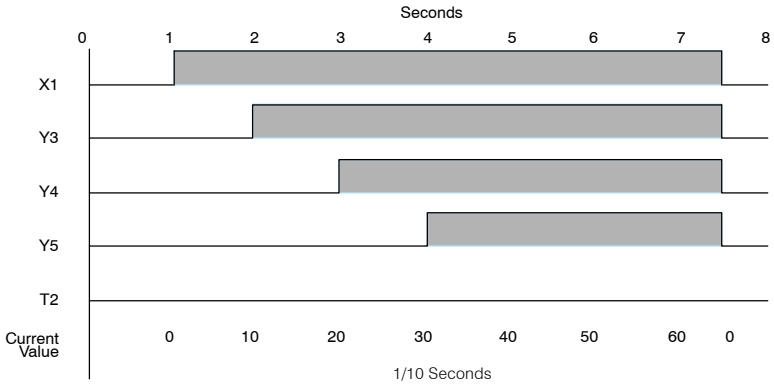
Timer Example Using Comparative Contacts

In the following example, a single input timer is used with a preset of 234.5 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.

DirectSOFT (see Note)



Timing Diagram



Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
TMR	TMR	2	0	K(CON)	2	3	4	5	↵
STR	TMR	2	0	K(CON)	1	0	↵		
OUT	Y(OUT)	3	↵						
STR	TMR	2	0	K(CON)	2	0	↵		
OUT	Y(OUT)	4	↵						
STR	TMR	2	0	K(CON)	3	0	↵		
OUT	Y(OUT)	5	↵						

NOTE: Since this representation is showing a ***Direct***SOFT example, you would use the alias TA20 (or V20) instead of T20, which would be necessary for the equivalent rung entered with the Handheld Programmer.

Accumulating Timer (TMRA) and Accumulating Fast Timer (TMRAF)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

The Accumulating Timer is a 0.1 second two input timer that times to a maximum of 9999999.9. The Accumulating Fast Timer is a 0.01 second two input timer that times to a maximum of 999999.99. These timers have two inputs, an enable and a reset. The timer will start timing when the enable is on and stop timing when the enable is off without resetting the current value to 0. The reset will reset the timer when on and allow the timer to time when off.

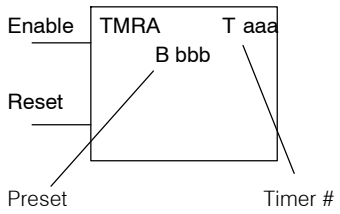
Instruction Specifications

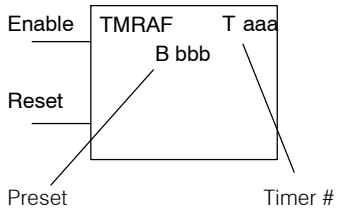
Timer Reference (Taaa): Specifies the timer number.

Preset Value (Bbbb): Constant value (K) or a V-memory location. (V locations are 32-bit double words.)

Current Value: Timer current value is a double word accessed by referencing the associated V or T memory location*. For example, the timer current value for T0 resides in V-memory location V0 and V1. (V locations are 16-bit words.)

Discrete Status Bit: The discrete status bit is accessed by referencing the associated T memory location. It will be on if the current value is equal to or greater than the preset value. For example the discrete status bit for timer 2 would be T2.





The timer discrete status bit and the current value is not specified in the timer instruction.

Caution: The TMRA uses two consecutive timer locations, since the preset can now be 8 digits, which requires two V-memory locations. For example, if TMRA T0 is used in the program, the next available timer is T2. Or if T0 was a normal timer, and T1 was an accumulating timer, then the next available timer would be T3.

Operand Data Type	B	DL430 Range		DL440 Range		DL450 Range	
		aaa	bbb	aaa	bbb	aaa	bbb
Timers	T	0-176	--	0-376	--	0-376	--
V-memory for preset values	V	--	1400-7377	--	1400-7377 10000-17777	--	1400-7377 10000-17777
Pointers (preset only)	P	--	--	--	1400-7377 10000-17777	--	1400-7377 10000-37777
Constants (preset only)	K	--	0-99999999	--	0-99999999	--	0-99999999
Timer discrete status bits	T	0-177		0-377		0-377	
Timer current values	V/T*	0-177		0-377		0-377	

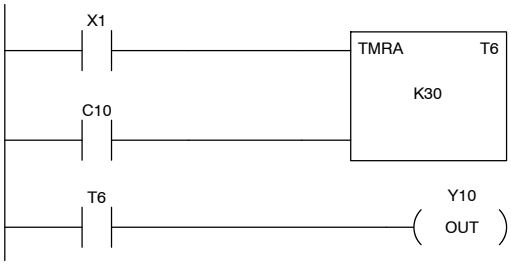
There are two methods of programming timers. You can perform functions when the timer reaches the specified preset using the the discrete status bit, or use the comparative contacts to perform functions at different time intervals based on one timer. The following examples show each method of using timers.

NOTE:* For the Handheld Programmer, both the Timer discrete status bits and current value can be accessed with the same data type (example T2). The way the data type is used determines if it is a status bit or a current value. Any comparative instruction using T2 will access the current value, all other instructions using T2 will access the status bit. Current values may also be accessed by the V-memory location. For ***Direct***SOFT, the use of T2 will refer to the timer's discrete status bit. You should use V2 (or the alias TA2) to refer to the current value.

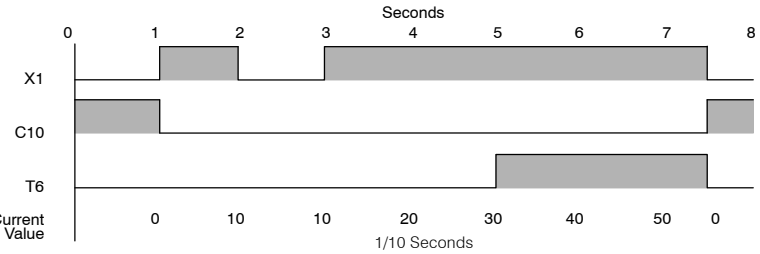
Accumulating Timer Example using Discrete Status Bits

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 the timer times for 1 second , stops for one second, then resumes timing. The timer will reset when C10 turn on after 5.5 seconds turning the discrete status bit off and resetting the timer current value to 0.

DirectSOFT



Timing Diagram



Handheld Programmer Keystrokes

STR	X(IN)	1	↵		
STR	C(CR)	1	0	↵	
TMR	SHFT	A	SHFT	TMR	6

Handheld Programmer Keystrokes (cont)

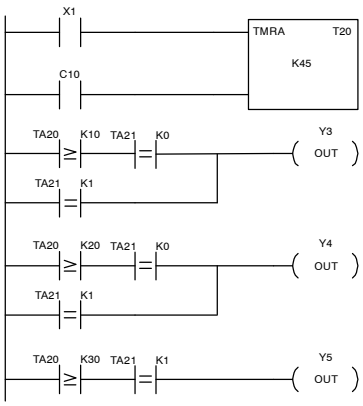
K(CON)	3	0	↵
STR	TMR	6	↵
OUT	Y(OUT)	1	0	↵

Accumulator Timer Example Using Comparative Contacts

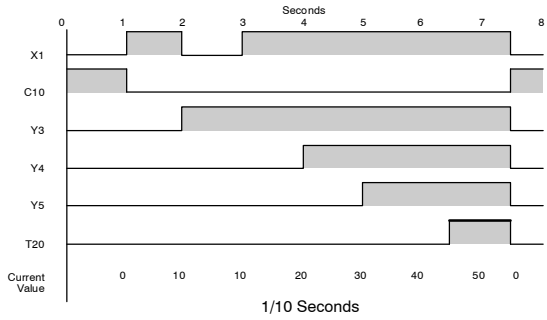
In the following example, a single input timer is used with a preset of 23458.9 seconds. Comparative contacts are used to energized Y3, Y4, and Y5 at one second intervals respectively. The comparative contacts will turn off when the timer is reset.

Contacts

Direct SOFT



Timing Diagram



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT							
$ STR	→	SHFT	C 2	B 1	A 0	ENT				
N TMR	SHFT	A 0	→	C 2	A 0	→	E 4	F 5	ENT	
$ STR	→	SHFT	T MLR	C 2	A 0	→	B 1	A 0	ENT	
V AND	SHFT	E 4	→	SHFT	T MLR	C 2	B 1	→	A 0	ENT
Q OR	SHFT	E 4	→	SHFT	T MLR	C 2	B 1	→	B 1	ENT
GX OUT	→	D 3	ENT							

Handheld Programmer Keystrokes (cont'd)

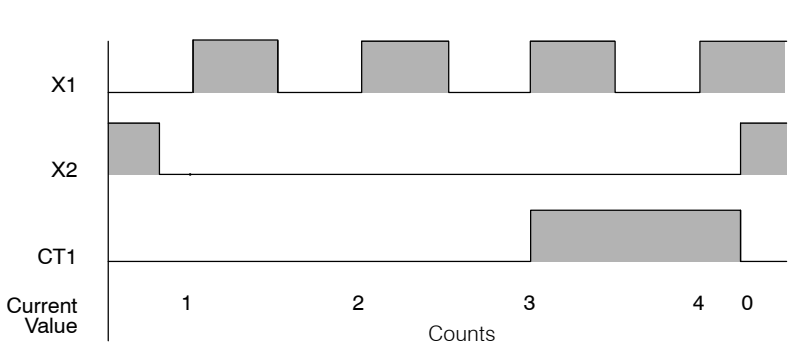
$ STR	→	SHFT	T MLR	C 2	A 0	→	C 2	A 0	ENT	
V AND	SHFT	E 4	→	SHFT	T MLR	C 2	B 1	→	A 0	ENT
Q OR	SHFT	E 4	→	SHFT	T MLR	C 2	B 1	→	B 1	ENT
GX OUT	→	E 4	ENT							
$ STR	→	SHFT	T MLR	C 2	A 0	→	D 3	A 0	ENT	
V AND	SHFT	E 4	→	SHFT	T MLR	C 2	B 1	→	B 1	ENT
GX OUT	→	F 5	ENT							

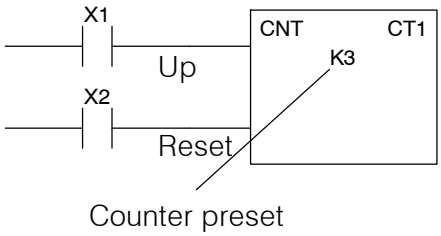
NOTE: Since this representation is showing a ***Direct***SOFT example, you would use the alias TA20 (or V20) instead of T20, which would be necessary for the equivalent rung entered with the Handheld Programmer.

Using Counters

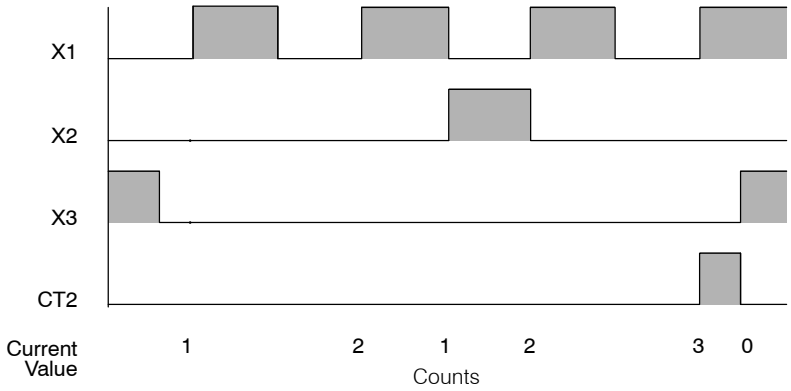
Counters are used to count events. The counters available are up counters, up/down counters, and stage counters (used with RLL*PLUS* programming).

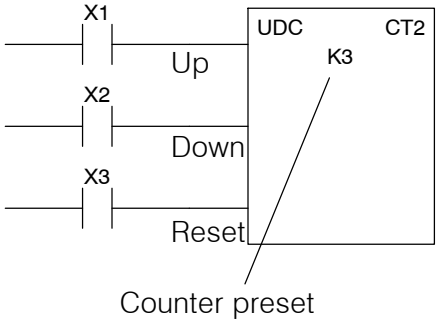
The up counter has two inputs, a count input and a reset input. The maximum count value is 9999. The timing diagram below shows the relationship between the counter input, counter reset, associated discrete bit, current value, and counter preset.



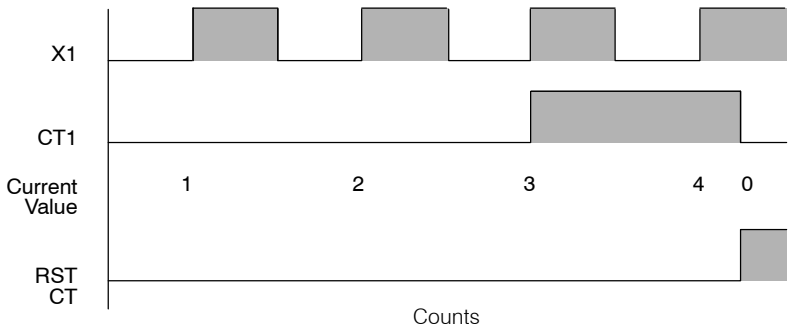


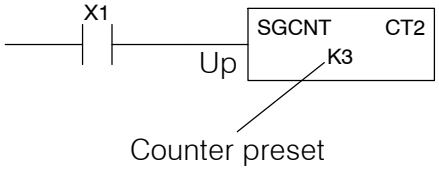
The up down counter has three inputs, a count up input, count down input and reset input. The maximum count value is 99999999. The timing diagram below shows the relationship between the counter input, counter reset, associated discrete bit, current value, and counter preset.





The stage counter has a count input and is reset by the RST instruction. This instruction is useful when programming RLL*PLUS*, by allowing you to reference the same counter from multiple stages. 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.



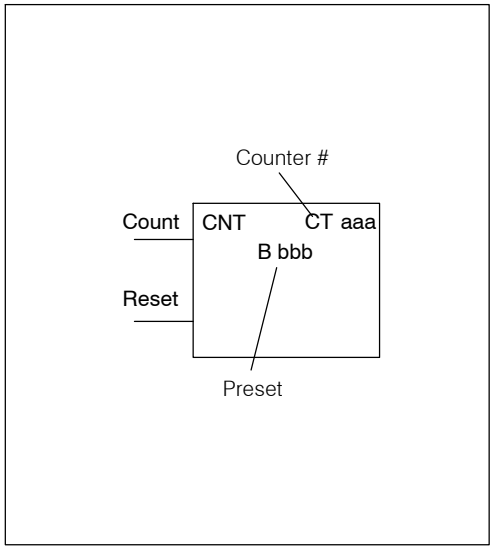


Counter (CNT)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

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.



The counter discrete status bit and the current value are not specified in the counter instruction.

Instruction Specifications

Counter Reference (CTaaa): Specifies the counter number.

Preset Value (Bbbb): Constant value (K) or a V-memory location. (V locations are 16-bit words.)

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. (V locations are 16-bit words.)

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.

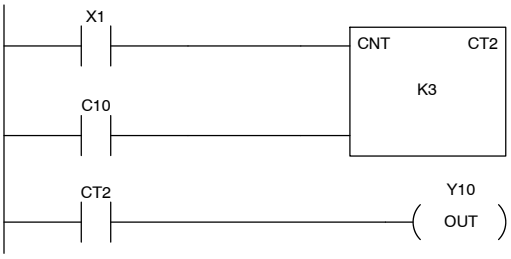
Operand Data Type		DL430 Range		DL440 Range		DL450 Range	
	B	aaa	bbb	aaa	bbb	aaa	bbb
Counters	CT	0-177	--	0-177	--	0-377	--
V-memory (preset only)	V	--	1400-7377	--	1400-7377 10000-17777	--	1400-7377 10000-37777
Pointers (preset only)	P	--	--	--	1400-7377 10000-17777	--	1400-7377 10000-37777
Constants (preset only)	K	--	0-9999	--	0-9999	--	0-9999
Counter discrete status bits	CT	0-177		0-177		0-377	
Counter current values	V/CT*	1000-1177		1000-1177		1000-1377	

NOTE:* For the Handheld Programmer, both the Counter discrete status bits and current value can be accessed with the same data type (example CT2). The way the data type is used determines if it is a status bit or a current value. Any comparative instruction using CT2 will access the current value, all other instructions using CT2 will access the status bit. Current values may also be accessed by the V-memory location. For ***Direct***SOFT, the use of CT2 will refer to the timer's discrete status bit. You should use V1002 (or the alias CTA2) to refer to the current value.

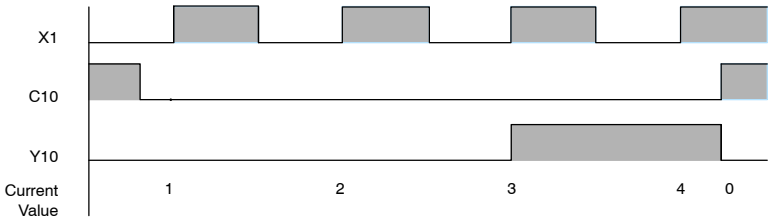
Counter Example Using Discrete Status Bits

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 Y10. 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.

DirectSOFT



Counting diagram



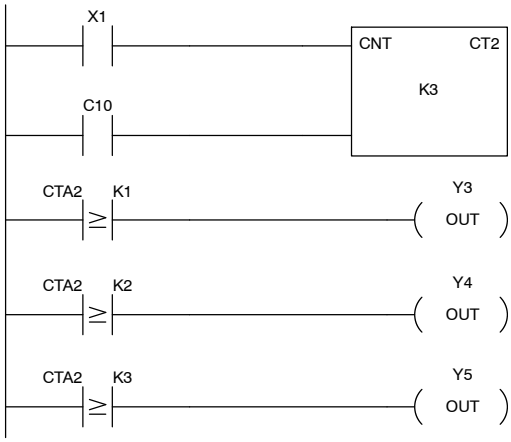
Handheld Programmer Keystrokes

STR	X(IN)	1	↵		
STR	C(CR)	1	0	↵	
CNT	CNT	2	K(CON)	3	↵
STR	CNT	2	↵		
OUT	Y(OUT)	1	0	↵	

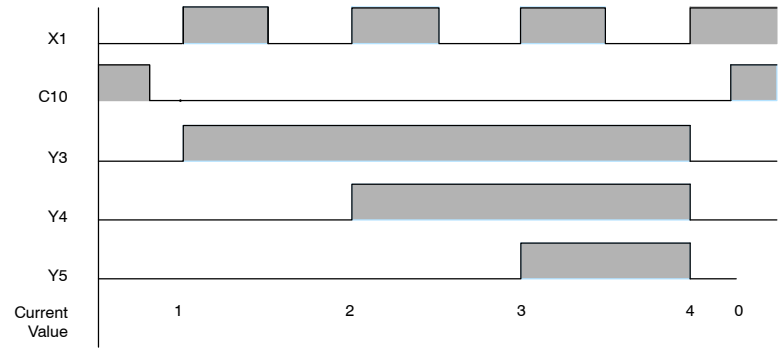
Counter Example Using Comparative Contacts

In the following example, when X1 is 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. The comparative contacts will turn off when the counter is reset. When the reset C10 turns on, the counter status bit will turn off and the counter current value will be 0.

DirectSOFT (see Note)



Counting diagram



Handheld Programmer Keystrokes

STR	X(IN)	1	↵		
STR	C(CR)	1	0	↵	
CNT	CNT	2	K(CON)	3	↵
STR	CNT	2	K(CON)	1	↵
OUT	Y(OUT)	3	↵		

Handheld Programmer Keystrokes (cont)

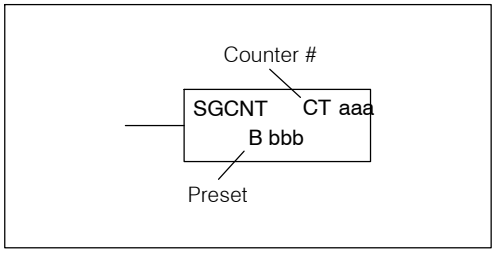
STR	CNT	2	K(CON)	2	↵
OUT	Y(OUT)	4	↵		
STR	CNT	2	K(CON)	3	↵
OUT	Y(OUT)	5	↵		

NOTE: Since this representation is showing a ***Direct***SOFT example, you would use the alias CTA2 (or V1002) instead of CT2, which would be necessary for the equivalent rung entered with the Handheld.

Stage Counter (SGCNT)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

The Stage Counter is a single input counter that increments when the input logic transitions from off to on. This counter differs from other counters since it will hold its current value until reset using the RST instruction. The Stage Counter is designed for use in **RLL***PLUS* programs but can be used in relay ladder logic programs. When the current value equals the preset value, the counter status bit turns on and the counter continues to count up to a maximum count of 9999. The maximum value will be held until the counter is reset.



The counter discrete status bit and the current value are not specified in the counter instruction.

Instruction Specifications

Counter Reference (CTaaa): Specifies the counter number.
Preset Value (Bbbb): Constant value (K) or a V–memory location. (V locations are 16-bit words.)
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. (V locations are 16-bit words.)
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.

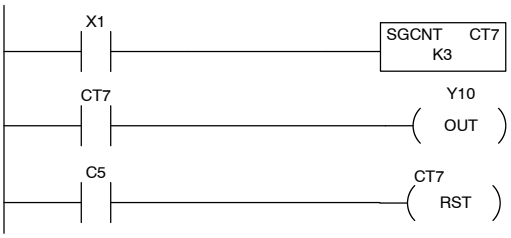
Operand Data Type		DL430 Range		DL440 Range		DL450 Range	
	B	aaa	bbb	aaa	bbb	aaa	bbb
Counters	CT	0–177	--	0–177	--	0–377	--
V–memory	V	--	1400–7377	--	1400–7377 10000–17777	--	1400–7377 10000–17777
Pointers (preset only)	P	--	--	--	1400–7377 10000–17777	--	1400–7377 10000–37777
Constants	K	--	0–9999	--	0–9999	--	0–9999
Counter discrete status bits	CT	0–177		0–177		0–377	
Counter current values	V/CT*	1000–1177		1000–1177		1000–1377	

NOTE:* For the Handheld Programmer, both the Stage Counter discrete status bits and current value can be accessed with the same data type (example CT2). The way the data type is used determines if it is a status bit or a current value. Any comparative instruction using CT2 will access the current value, all other instructions using CT2 will access the status bit. Current values may also be accessed by the V-memory location. For ***Direct***SOFT, the use of CT2 will refer to the timer's discrete status bit. You should use V1002 (or the alias CTA2) to refer to the current value.

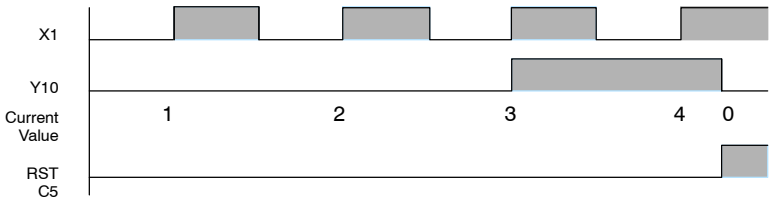
Stage Counter Example Using Discrete Status Bits

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 Y10. 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.

DirectSOFT



Counting diagram



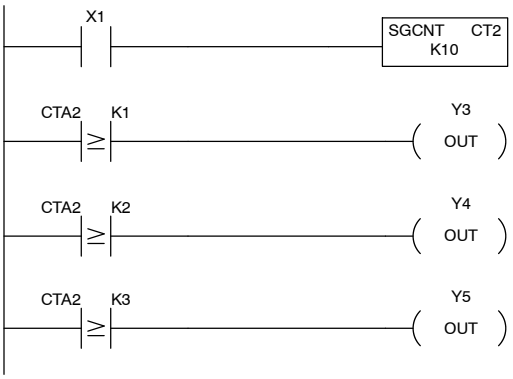
Handheld Programmer Keystrokes

STR	X(IN)	1	↵			
SG	CNT	CNT	7	K(CON)	3	↵
STR	CNT	7	↵			
OUT	Y(OUT)	1	0	↵		
STR	C(CR)	5	↵			
RST	CNT	7	↵			

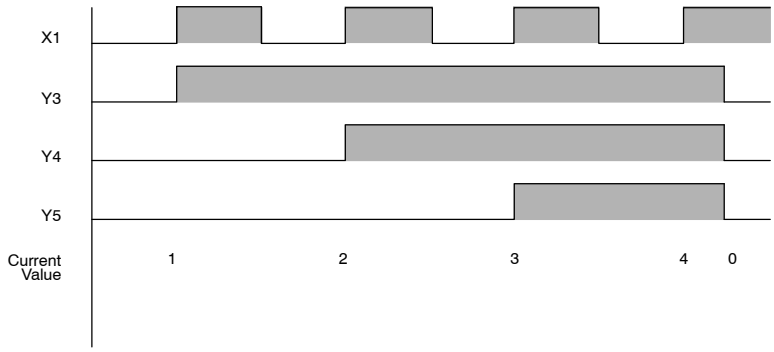
Stage Counter Example Using Comparative Contacts

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.

tSOFT (see Note)



Counting diagram



Handheld Programmer Keystrokes

STR	X(IN)	1	↵				
SG	CNT	CNT	2	K(CON)	1	0	↵
STR	CNT	2	K(CON)	1	↵		
OUT	Y(OUT)	3	↵				

Handheld Programmer Keystrokes (cont)

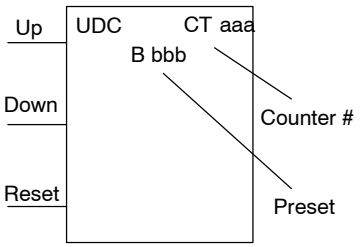
STR	CNT	2	K(CON)	2	↵
OUT	Y(OUT)	4	↵		
STR	CNT	2	K(CON)	3	↵
OUT	Y(OUT)	5	↵		

NOTE: Since this representation is showing a ***Direct***SOFT example, you would use the alias CTA2 (or V1002) instead of CT2, which would be necessary for the equivalent rung entered with the Handheld.

Up/Down Counter (UDC)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

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.



The counter discrete status bit and the current value is not specified in the counter instruction.

Caution: The UDC uses two consecutive counter locations, since the preset can now be 8 digits, which requires two V-memory locations. For example, if UDC CT0 is used in the program, the next available counter is CT2. Or if CT0 was a normal counter, and CT1 was an up/down counter, then the next available counter would be CT3.

Instruction Specification

Counter Reference (CTaaa): Specifies the counter number.
Preset Value (Bbbb): Constant value (K) or two consecutive V-memory locations. (V locations are 16-bit words.)
Current Values: Current count is a double word value accessed by referencing the associated V or CT memory locations*. The V-memory location is the counter location + 1000. For example, the counter current value for CT5 resides in V-memory location V1005 and V1006. (V locations are 16-bit words.)
Discrete Status Bit: The discrete status bit is accessed by referencing the associated CT memory location. It will be on if value is equal to or greater than the preset value. For example the discrete status bit for counter 2 would be CT2.

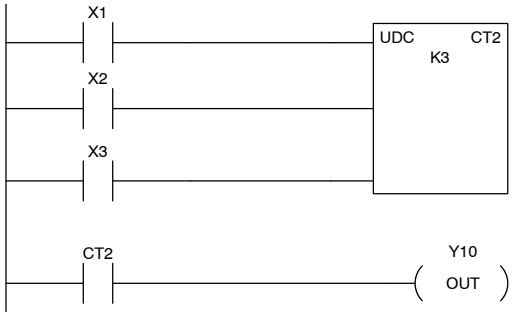
Operand Data Type		DL430 Range		DL440 Range		DL450 Range	
	B	aaa	bbb	aaa	bbb	aaa	bbb
Counters	CT	0-176	--	0-176	--	0-376	--
V-memory	V	--	1400-7377	--	1400-7377 10000-17777	--	1400-7377 10000-37777
Pointers (preset only)	P	--	--	--	1400-7377 10000-17777	--	1400-7377 10000-37777
Constants	K	--	0-99999999	--	0-99999999	--	0-99999999
Counter discrete status bits	CT	0-177		0-177		0-377	
Counter current values	V/CT*	1000-1177		1000-1177		1000-1377	

NOTE:* For the Handheld Programmer, both the Stage Counter discrete status bits and current value can be accessed with the same data type (example CT2). The way the data type is used determines if it is a status bit or a current value. Any comparative instruction using CT2 will access the current value, all other instructions using CT2 will access the status bit. Current values may also be accessed by the V-memory location. For ***Direct***SOFT, the use of CT2 will refer to the timer's discrete status bit. You should use V1002 (or the alias CTA2) to refer to the current value.

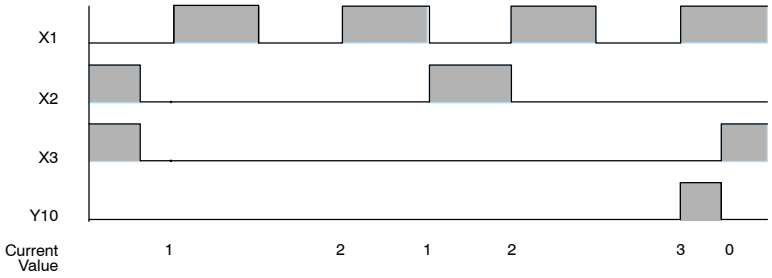
Up/Down Counter Example Using Discrete Status Bits

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

DirectSOFT



Counting Diagram



Handheld Programmer Keystrokes

STR	X(IN)	1	↵			
STR	X(IN)	2	↵			
STR	X(IN)	3	↵			
SHFT	U	D	C	SHFT	CNT	2

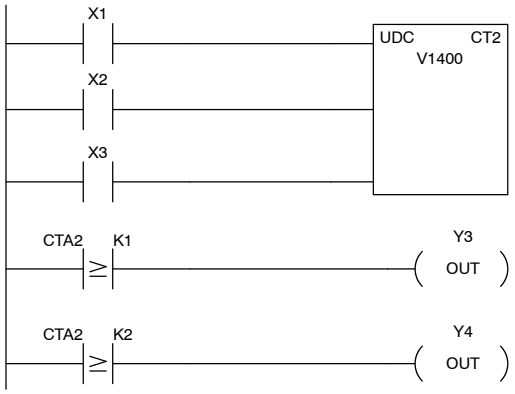
Handheld Programmer Keystrokes (cont)

K(CON)	3	↵		
STR	CNT	2	↵	
OUT	Y(OUT)	1	0	↵

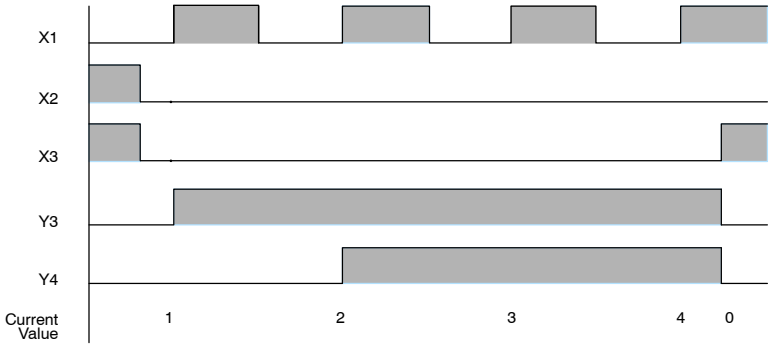
Up/Down Counter Example Using Comparative Contacts

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. The comparative contacts will turn off when the counter is reset. When the reset X3 turns on, the counter status bit will turn off and the current value will be 0.

DirectSOFT (see Note)



Counting Diagram



Handheld Programmer Keystrokes

STR	X(IN)	1	↵			
STR	X(IN)	2	↵			
STR	X(IN)	3	↵			
SHFT	U	D	C	SHFT	CNT	2
V	1	4	0	0	↵	
STR	CNT	2	K(CON)	1	↵	

Handheld Programmer Keystrokes (cont)

OUT	Y(OUT)	3	↵			
STR	CNT	2	K(CON)	2	↵	
OUT	Y(OUT)	4	↵			

NOTE: Since this representation is showing a ***Direct***SOFT example, you would use the alias CTA2 (or V1002) instead of CT2, which would be necessary for the equivalent rung entered with the Handheld.

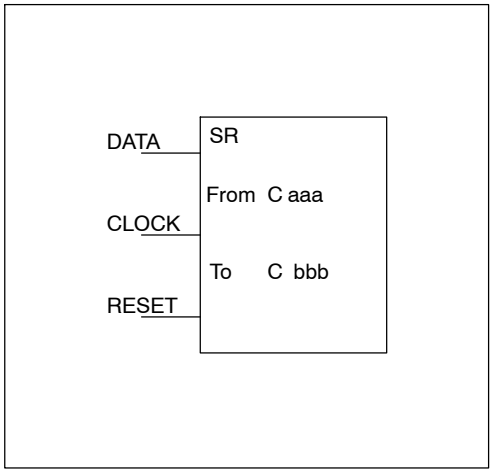
Shift Register (SR)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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 end at the end of an 8 bit boundary.

The Shift Register has three contacts.

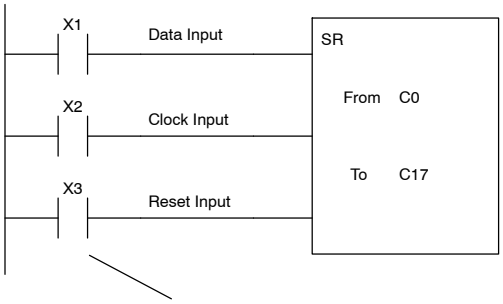
- Data — determines the value (1 or 0) that will enter the register
- Clock — shifts the bits one position on each low to high transition
- Reset —resets the Shift Register to all zeros.



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 lower address to higher address. From C17 to C0 would define a block of sixteen bits, to be shifted from higher address to lower address. 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		DL430 Range		DL440 Range		DL440 Range	
	B	aaa	bbb	aaa	bbb	aaa	bbb
Control Relay	C	0–737	0–737	0–1777	0–1777	0–3777	0–3777

DirectSOFT



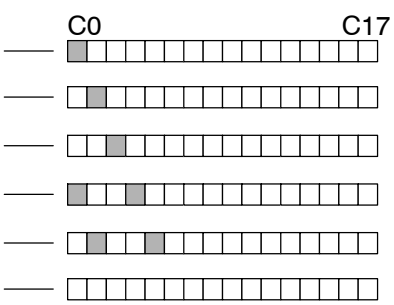
Handheld Programmer Keystrokes

STR	X(IN)	1	↵
STR	X(IN)	2	↵
STR	X(IN)	3	↵
SR	C(CR)	0	
C(CR)	1	7	↵

Inputs on Successive Scans

Data	Clock	Reset
1	⌐¬	0
0	⌐¬	0
0	⌐¬	0
1	⌐¬	0
0	⌐¬	0
–	–	1

Shift Register Bits



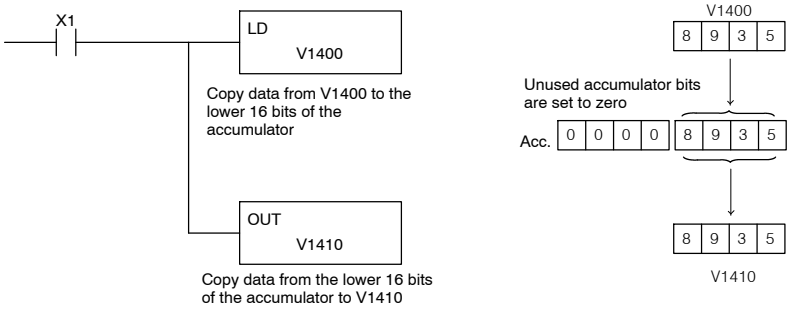
Accumulator/Data Stack Load and Output Instructions

Using the Accumulator

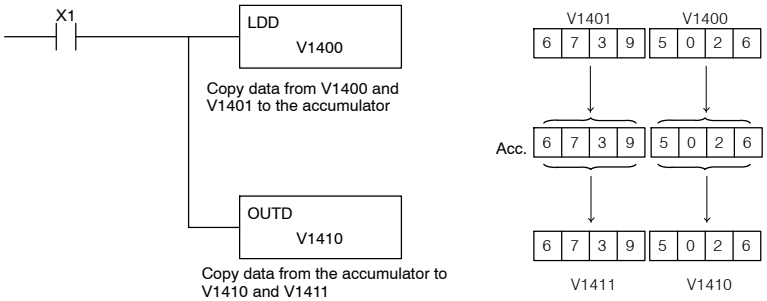
The accumulator in the DL405 series CPUs is a 32 bit register which is used as a temporary storage location for data that is being copied or manipulated in some manner. For example, you have to use the accumulator to perform math operations such as add, subtract, multiply, etc. Since there are 32 bits, you can use up to an 8-digit BCD number, or a 32-bit 2's complement number. The accumulator is reset to 0 at the end of every CPU scan.

Copying Data to the Accumulator

The Load and Out instructions and their variations are used to copy data from a V-memory location to the accumulator, or, to copy data from the accumulator to V-memory. The following example copies data from V-memory location V1400 to V-memory location V1410.

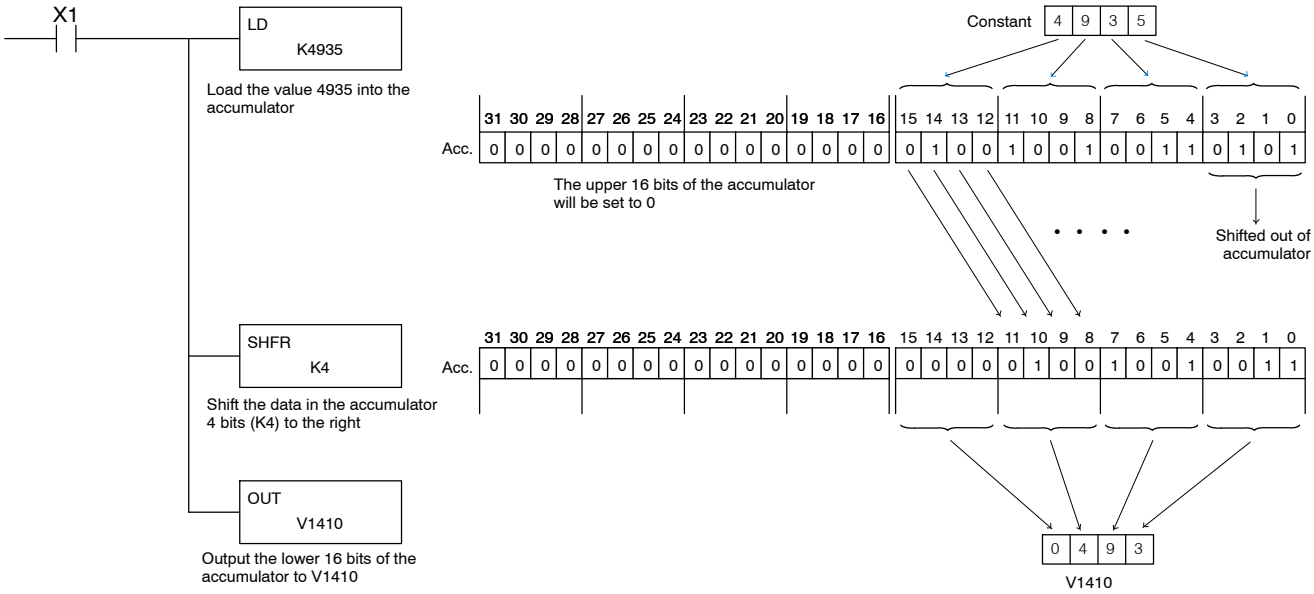


Since the accumulator is 32 bits and V-memory locations are 16 bits, the Load Double and Out Double (or variations thereof) use two consecutive V-memory locations or 8 digit BCD constants to copy data either to the accumulator from a V-memory address or from a V-memory address to the accumulator. For example, if you wanted to copy data from V-memory location V1400 and V1401 to V-memory location V1410 and V1411, the most efficient way to perform this function would be as follows:

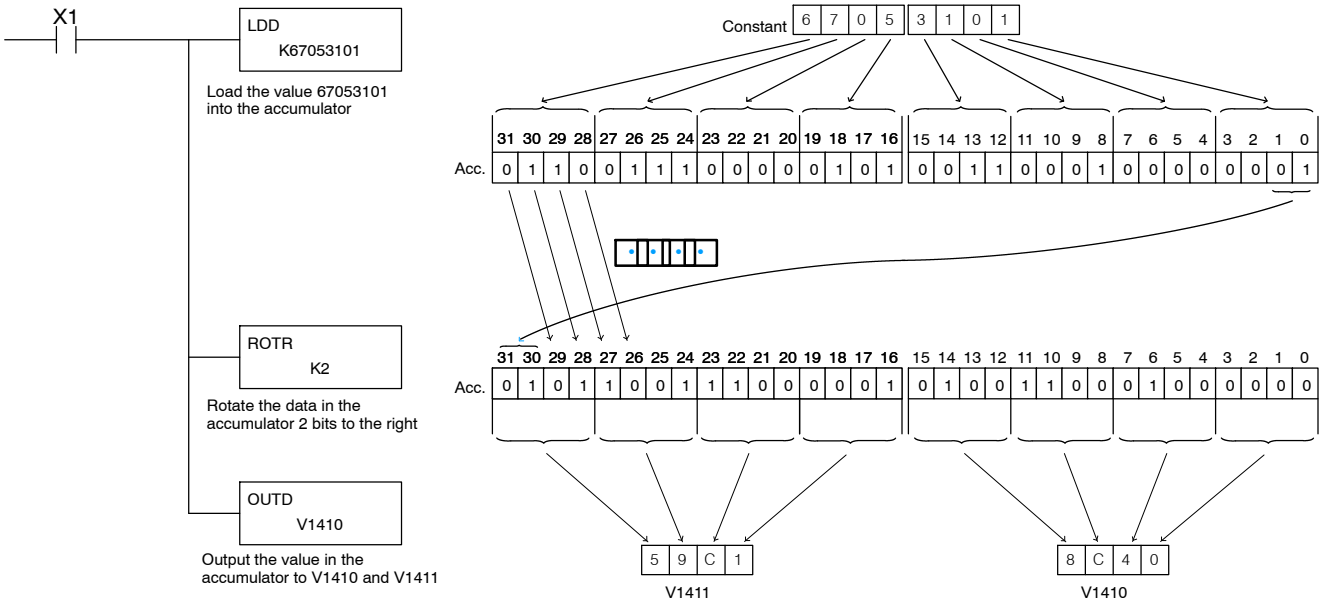


Changing the Accumulator Data

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 BCD value 4935 into the accumulator, shifts the data right 4 bits, and outputs the result to V1410.

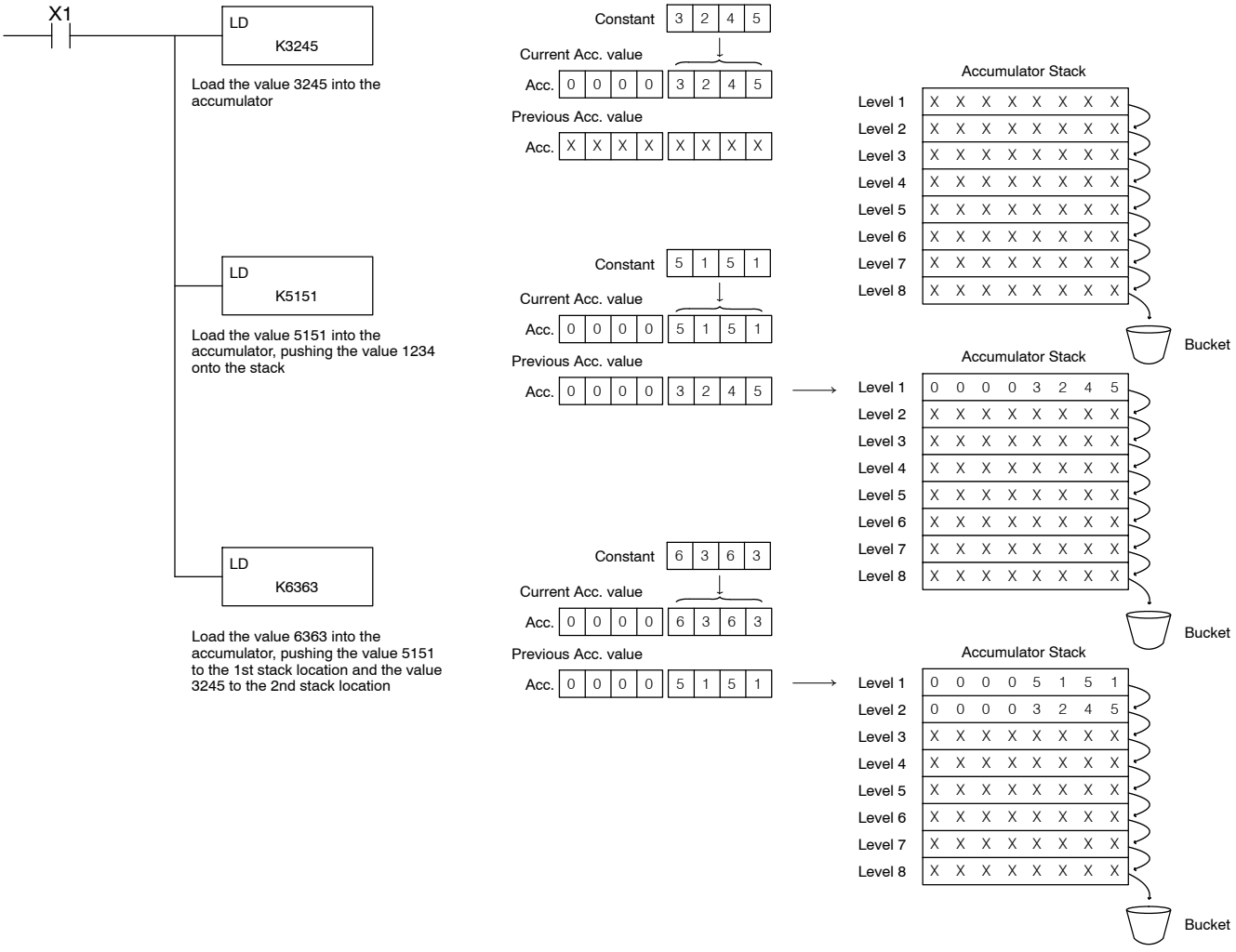


Some of the data manipulation instructions use 32 bits. They use two consecutive V-memory locations or 8 digit BCD constants to manipulate data in the accumulator. The following example rotates the value 67053101 two bits to the right and outputs the value to V1410 and V1411.

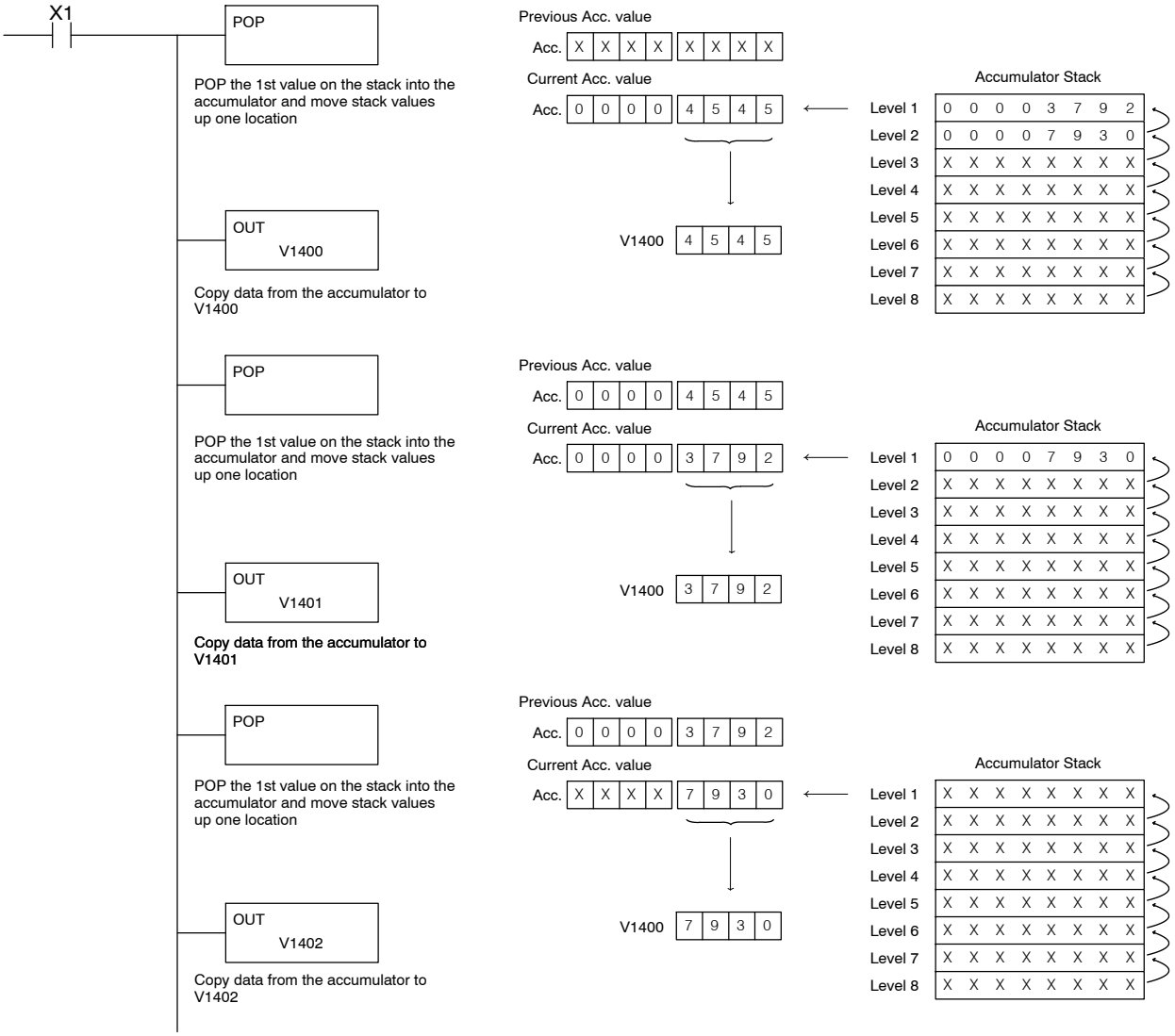


Using the Accumulator Stack

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 type instruction is executed without the use of the Out type instruction. The first load instruction in the scan places a value into the accumulator. Every Load instruction thereafter without the use of an Out instruction places a value into the accumulator and the value that was in the accumulator is placed onto the accumulator stack. The Out instruction nullifies the previous load instruction and does not place the value that was in the accumulator onto the accumulator stack when the next load instruction is executed. Every time a value is placed onto the accumulator stack the other values in the stack are pushed down one location. The accumulator is eight levels deep (eight 32 bit registers). If there is a value in the eighth location when a new value is placed onto the stack, the value in the eighth location is pushed off the stack and cannot be recovered.



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



Accumulator and Accumulator Stack Memory Locations

430	440	450
✗	✓	✓

DS	HPP
✓	✗

There may be times when you want to read a value that has 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
V701				V700			

Accumulator Stack

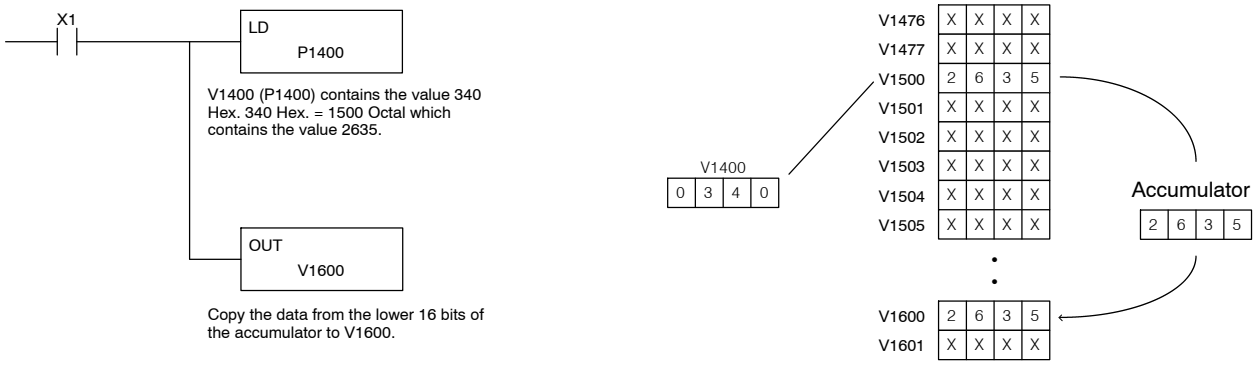
Level 1	0	0	0	0	3	7	9	2	V703 - V702
Level 2	0	0	0	0	7	9	3	0	V705 - V704
Level 3	X	X	X	X	X	X	X	X	V707 - V706
Level 4	X	X	X	X	X	X	X	X	V711 - V710
Level 5	X	X	X	X	X	X	X	X	V713 - V712
Level 6	X	X	X	X	X	X	X	X	V715 - V714
Level 7	X	X	X	X	X	X	X	X	V717 - V716
Level 8	X	X	X	X	X	X	X	X	V721 - V720

Using Pointers

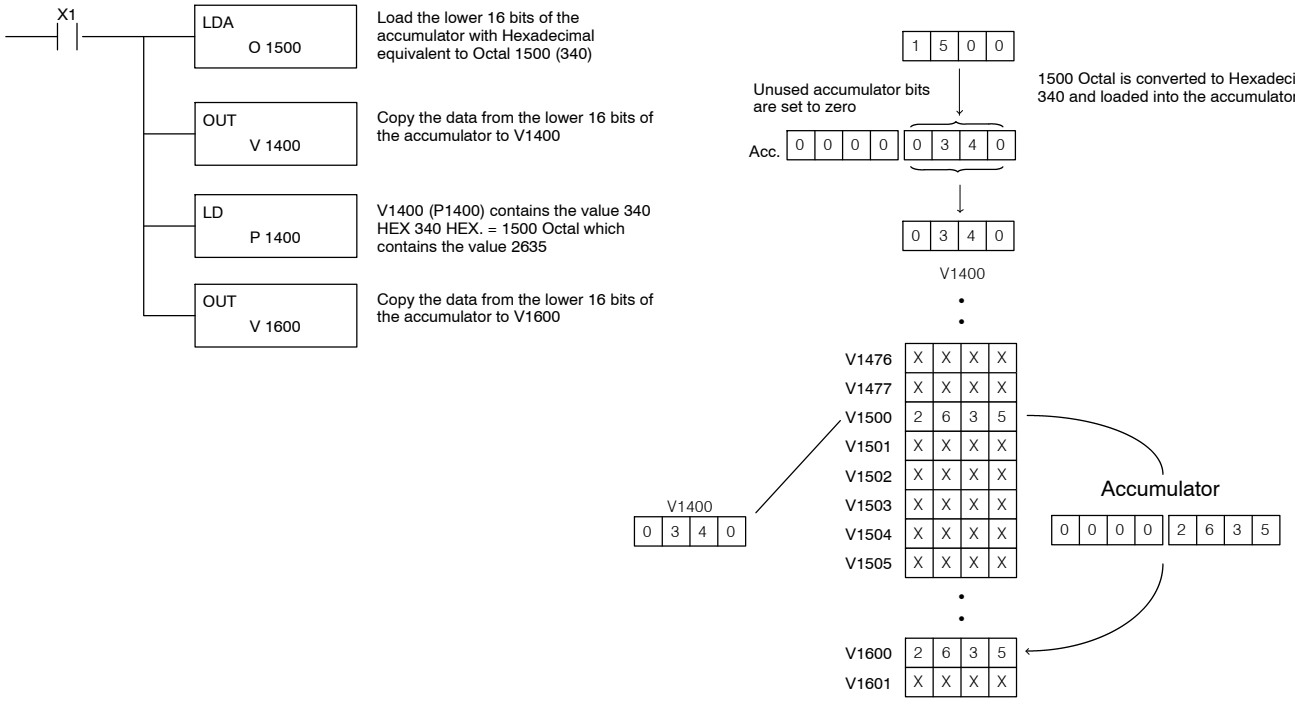
Many of the DL405 series instructions will allow V-memory pointers as a operand. Pointers can be useful in ladder logic programming, but can be difficult to understand or implement in your application if you do not have prior experience with pointers (commonly known as indirect addressing). Pointers allow instructions to obtain data from V-memory locations referenced by the pointer value.

NOTE: In the DL405 V-memory addressing is in octal. However the value in the pointer location which will reference a V-memory location is viewed as HEX. Use the Load Address instruction to move a address into the pointer location. This instruction performs the Octal to Hexadecimal conversion for you.

The following example uses a pointer operand in a Load instruction. V-memory location 1400 is the pointer location. V1400 contains the value 340 which is the HEX equivalent of the Octal address V-memory location V1500. The CPU copies the data from V1500 (contains 2635) into the lower word of the accumulator.



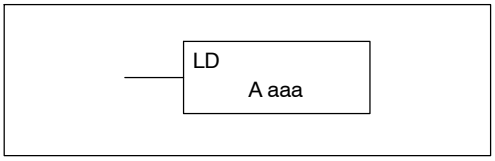
The following example is similar to the one above, except for the LDA (load address) instruction which automatically converts the Octal address to the Hex equivalent.



Load (LD)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

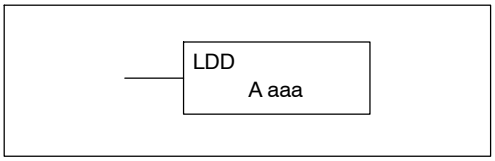
The Load instruction is a 16 bit instruction that loads the value (Aaaa) (either a V-memory location or a 4 digit constant) into the lower 16 accumulator bits. The upper 16 accumulator bits are set to 0.



Load Double (LDD)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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.

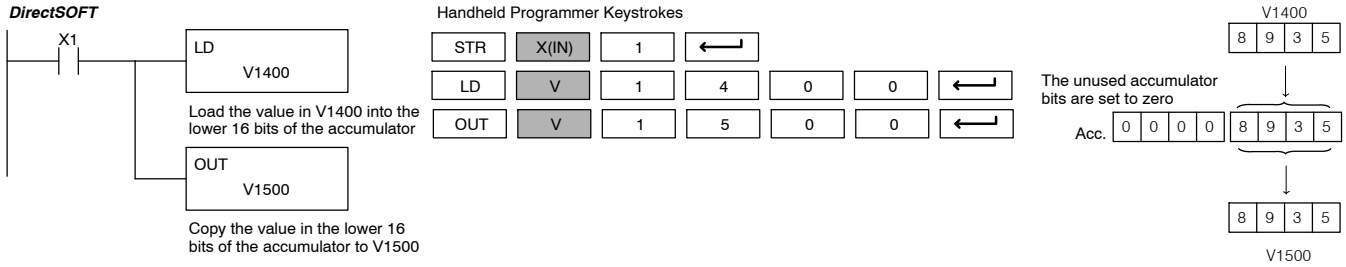


Operand Data Type		DL430 Range	DL440 Range	DL440 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All V mem (See p. 3-40)	All V mem (See p. 3-41)	All V mem (See p. 3-42)
Constant	K	0-FFFF	0-FFFF	0-FFFF

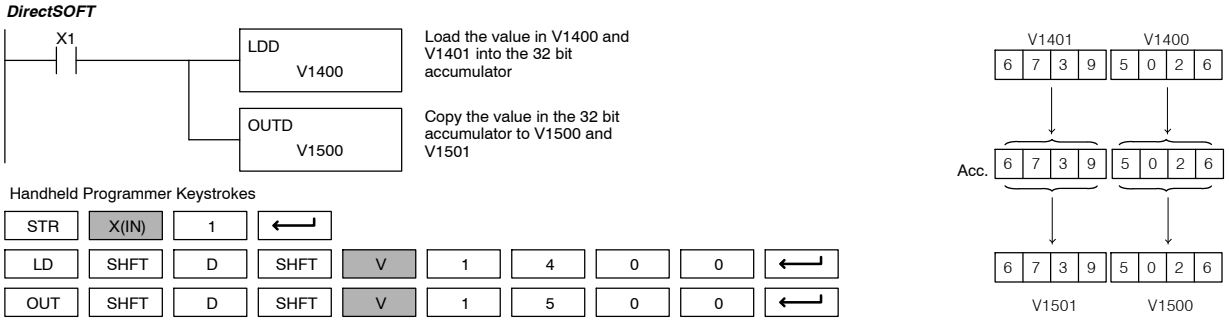
Discrete Bit Flags	Description
SP76	on when the value loaded into the accumulator by any instruction is zero.

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

In the following Load example, when X1 is on, the value in V1400 will be loaded into the accumulator and output to V1500.



In the following example, when X1 is on, the 32 bit value in V1400 and V1401 will be loaded into the accumulator and output to V1500 and V1501.

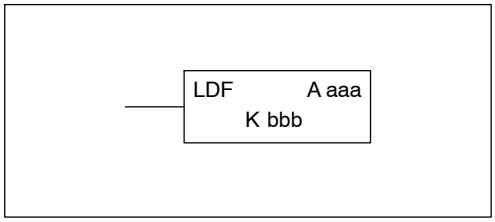


Load Formatted (LDF)

430	440	450
✓		✓

DS	HPP
✓	✓

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.



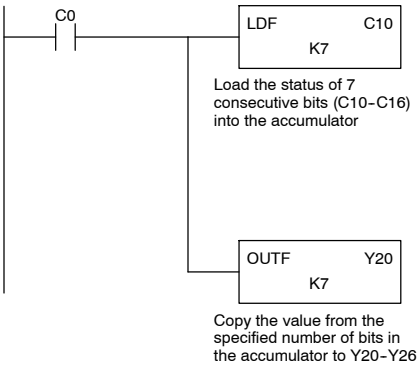
Operand Data Type		DL440 Range		DL450 Range	
	A	aaa	bbb	aaa	bbb
Inputs	X	0-477	--	0-1777	--
Outputs	Y	0-477	--	0-1777	--
Control Relays	C	0-1777	--	0-3777	--
Stage Bits	S	0-1777	--	0-1777	--
Timer Bits	T	0-377	--	0-377	--
Counter Bits	CT	0-177	--	0-377	--
Special Relays	SP	0-137 320-717	--	0-137 320-717	--
Global I/O	GX	0-1777	--	0-2777	--
Constant	K	--	1-32	--	1-32

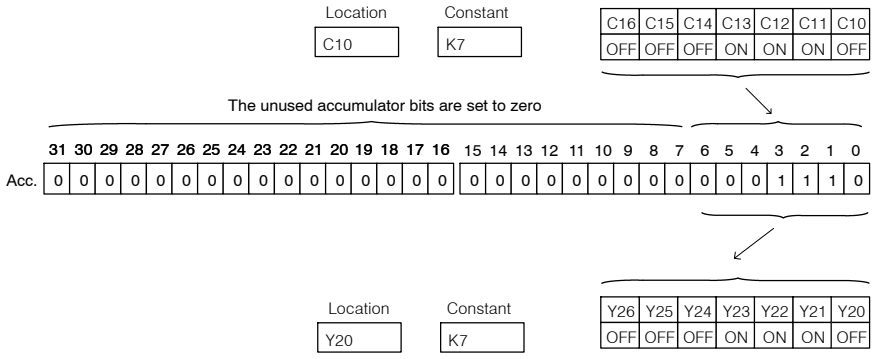
Discrete Bit Flags	Description
SP76	on when the value loaded into the accumulator by any instruction is zero.

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

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 6 bits of the accumulator are output to Y20-Y26 using the Out Formatted instruction.

DirectSOFT





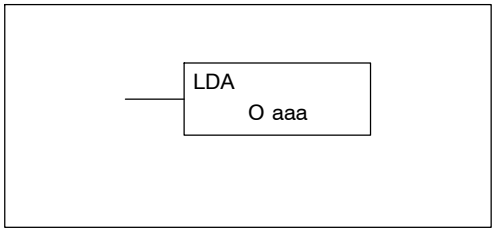
Handheld Programmer Keystrokes

STR	C(CR)	0	↵						
LD	SHFT	F	SHFT	C(CR)	1	0	K(CON)	7	↵
OUT	SHFT	F	SHFT	Y(OUT)	2	0	K(CON)	7	↵

Load Address (LDA)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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 DL405 system are in octal.

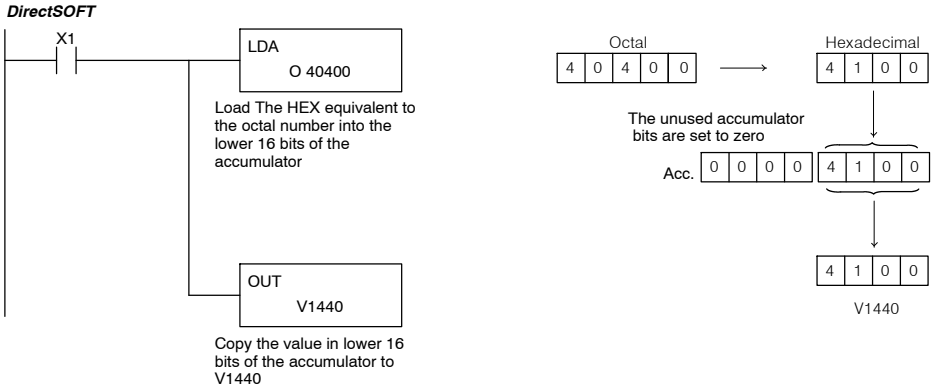


Operand Data Type	DL430 Range	DL440 Range	DL450 Range
	aaa	aaa	aaa
Octal Address O	0 – 77777	0 – 77777	0 – 177777

Discrete Bit Flags	Description
SP76	on when the value loaded into the accumulator by any instruction is zero.

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

In the following example when X1 is on, the octal number 40400 will be converted to a HEX 4100 and loaded into the accumulator using the Load Address instruction. The value in the lower 16 bits of the accumulator is copied to V1440 using the Out instruction.



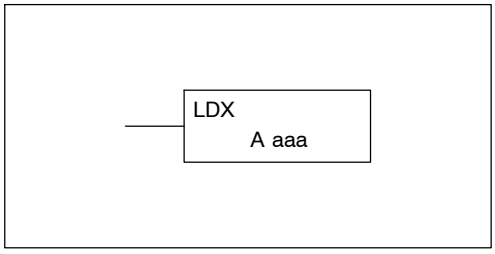
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	A	OCT	4	0	4	0	0	↵
OUT	V	1	4	4	0	↵			

Load Accumulator Indexed (LDX)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

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.



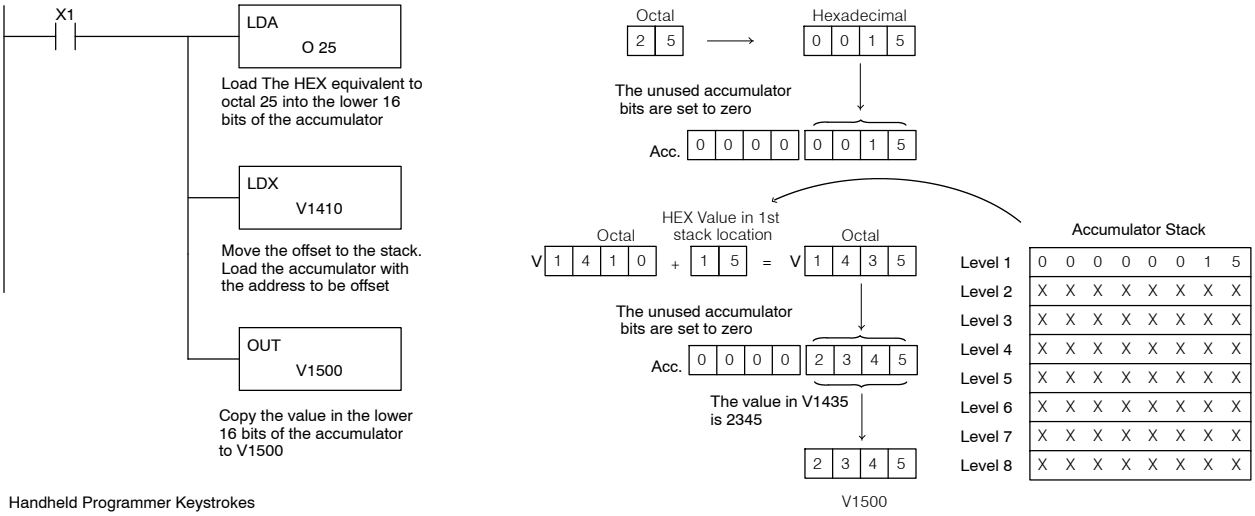
Helpful Hint: — The Load Address instruction can be used to convert an octal address to a HEX address and load the value into the accumulator.

Operand Data Type		DL430 Range	DL440 Range	DL440 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	--	All (See p. 3-41)	All (See p. 3-42)

Discrete Bit Flags	Description
SP76	on when the value loaded into the accumulator by any instruction is zero.

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

In the following example when X1 is on, the 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 1st. 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.

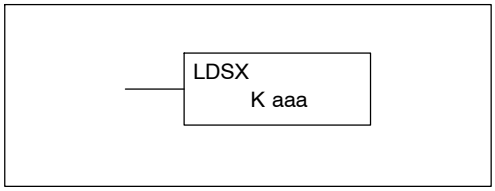


Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	A	OCT	2	5	↵			
LD	SHFT	X	SHFT	V	1	4	1	0	↵
OUT	V	1	5	0	0	↵			

Load Accumulator Indexed from Data Constants (LDSX)

430	440	450	DS	HPP
✗	✓	✓	✓	✓



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 accumulator's lower 16 bits.

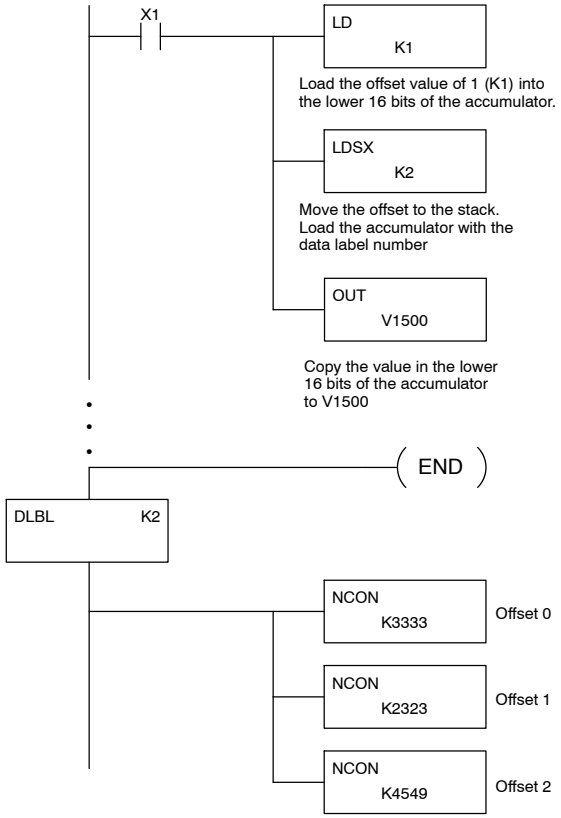
The LDSX instruction uses the value in the first level of the accumulator stack as an offset to determine which numerical or ASCII constant within the Data Label Area will be loaded into the accumulator. The LDSX instruction interprets the value in the first level of the accumulator stack as a HEX value.

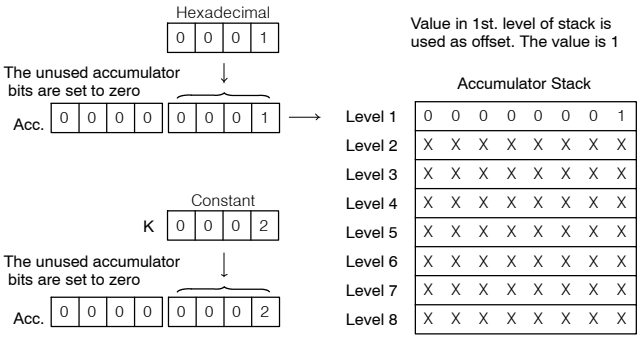
Helpful Hint: — The Load Address instruction can be used to convert octal to HEX and load the value into the accumulator.

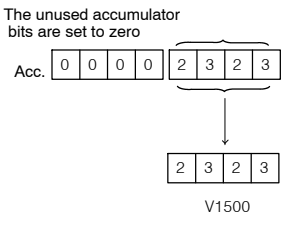
Operand Data Type		DL440 Range	DL450 Range
		aaa	aaa
Constant	K	1–FFFF	1–FFFF

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 Load instruction loads the offset of 1 into the accumulator. When the LDSX instruction executes, this value is placed into the first level of the accumulator stack. The LDSX instruction specifies the Data Label (DLBL K2) where the numerical constant(s) are located in the program. It loads the constant value according to the offset value into the accumulator's lower 16 bits.







Handheld Programmer Keystrokes

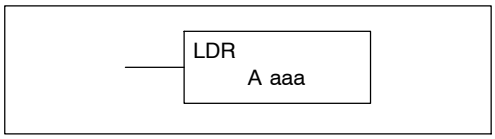
STR	X(IN)	1	↵								
LD	K(CON)	1	↵								
LD	SHFT	S	X	K(CON)	2	↵					
OUT	V	1	5	0	0	↵					
END	↵										
D	L	B	L	K(CON)	2	↵					
SHFT	N	C	O	N	SHFT	K(CON)	2	2	2	2	↵
SHFT	N	C	O	N	SHFT	K(CON)	2	3	2	3	↵
SHFT	N	C	O	N	SHFT	K(CON)	4	4	4	4	↵

Load Real Number (LDR)

430	440	450
✓	✓	✓

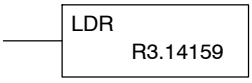
DS	HPP
✓	✓

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

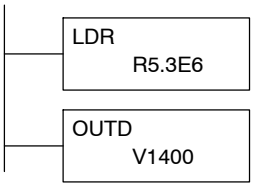


Operand Data Type		DL450 Range
	A	**aaa**
V-memory	V	All V mem (See p. 3-42)
Pointer	P	All V mem (See p. 3-42)
Real Constant	R	Full IEEE 32-bit range

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".

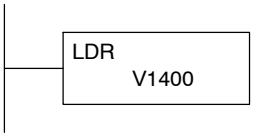


For very large numbers or very small numbers, you can use exponential notation. The number to the right is 5.3 million. The OUTD instruction stores it in V1400 and V1401.



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.

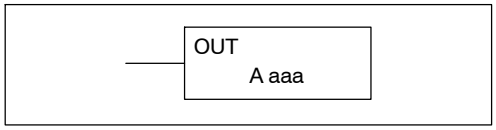
The previous example above stored a real number in V1400 and V1401. Suppose that now we want to retrieve that number. Just use the Load Real with the V data type, as shown to the right. Next we could perform real math on it, or convert it to a binary number.



Out (OUT)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

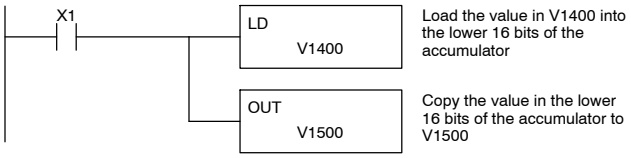
The Out instruction is a 16 bit instruction that copies the value in the lower 16 bits of the accumulator to a specified V-memory location (Aaaa).

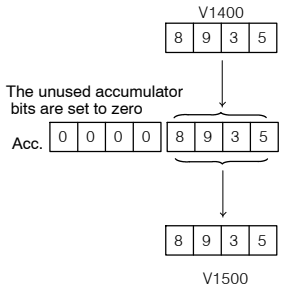


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)

In the following example, when X1 is on, the value in V1400 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the lower 16 bits of the accumulator is copied to V1500 using the Out instruction.

DirectSOFT





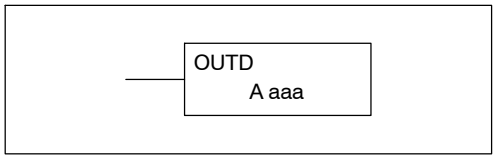
Handheld Programmer Keystrokes

STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
OUT	V	1	5	0	0	↵

Out DOUBLE (OUTD)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

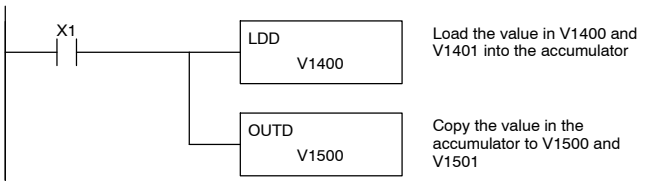
The Out Double instruction is a 32 bit instruction that copies the value in the accumulator to two consecutive V-memory locations at a Specified starting location (Aaaa).

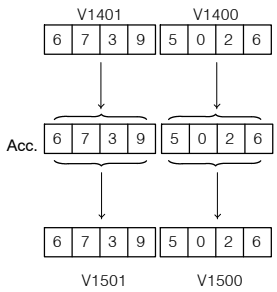


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All V mem (See p. 3-40)	All V mem (See p. 3-41)	All V mem (See p. 3-42)

In the following example, when X1 is on, the 32 bit value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is output to V1500 and V1501 using the Out Double instruction.

DirectSOFT





Handheld Programmer Keystrokes

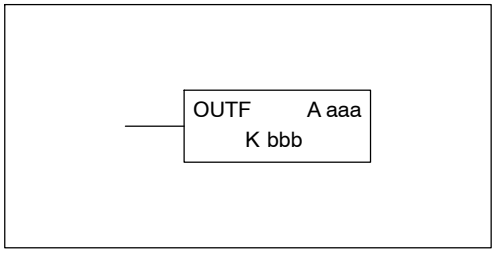
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Out Formatted (OUTF)

430	440	450
✗	✓	✓

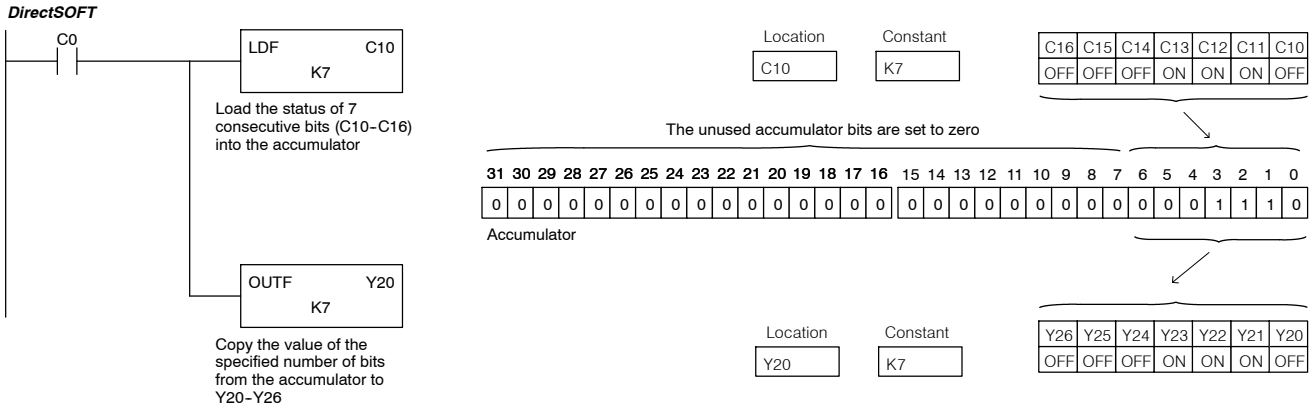
DS	HPP
✓	✓

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		DL440 Range		DL450 Range	
	A	aaa	bbb	aaa	bbb
Inputs	X	0-477	--	0-1777	--
Outputs	Y	0-477	--	0-1777	--
Control Relays	C	0-1777	--	0-3777	--
Global I/O	GX	0-1777	--	0-2777	--
Constant	K	--	1-32	--	1-32

In the following example, when C0 is on, the binary pattern of C10-C16 (7 bits) will be loaded into the accumulator using the Load Formatted instruction. The lower 7 bits of the accumulator are output to Y20-Y26 using the Out Formatted instruction.



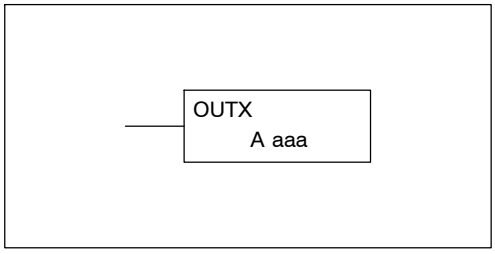
Handheld Programmer Keystrokes

STR	C(CR)	0	↵					
LD	SHFT	F	SHFT	C(CR)	1	0	K(CON)	7	↵
OUT	SHFT	F	SHFT	Y(OUT)	2	0	K(CON)	7	↵

Out Indexed (OUTX)

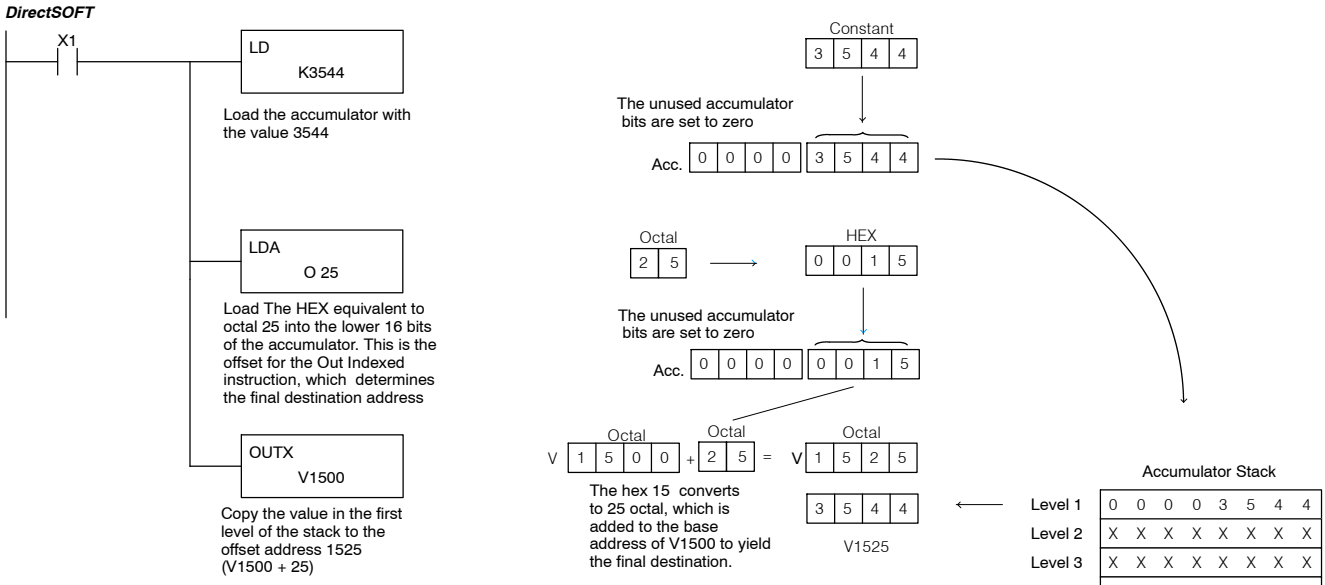
✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

The Out Indexed instruction is a 16 bit instruction. It copies a 16 bit or 4 digit value from the first level of the accumulator stack to a source address offset by the value in the accumulator(V-memory + offset).This instruction interprets the offset value as a HEX number. The upper 16 bits of the accumulator are set to zero.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)

In the following example, when X1 is on, the constant value 3544 is loaded into the accumulator. This is the value that will be output to the specified offset V-memory location (V1525). The value 3544 will be placed onto the stack when the Load Address instruction is executed. Remember, two consecutive Load instructions places the value of the first load instruction onto the stack. The Load Address instruction converts octal 25 to HEX 15 and places the value in the accumulator. The Out Indexed instruction outputs the value 3544 which resides in the first level of the accumulator stack to V1525.



Handheld Programmer Keystrokes

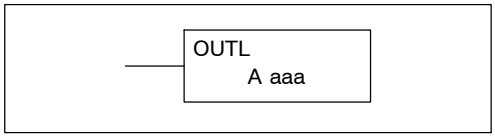
STR	X(IN)	1	↵					
LD	K	3	5	4	4	↵		
LD	SHFT	A	OCT	2	5	↵		
OUT	SHFT	X	SHFT	V	1	5	0	0	↵

Out Least (OUTL)

430	440	450
✗	✗	✓

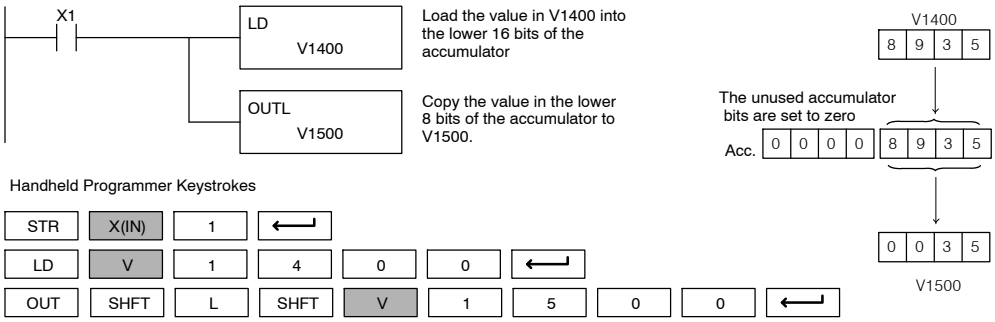
DS	HPP
✓	✓

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



In the following example, when X1 is on, the value in V1400 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the lower 8 bits of the accumulator are copied to V1500 using the Out Least instruction.

DirectSOFT



Handheld Programmer Keystrokes

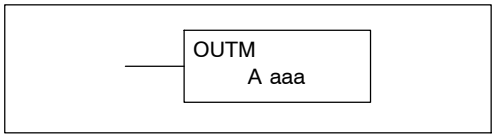
STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
OUT	SHFT	L	SHFT	V	1	5	0	0	↵

Out Most (OUTM)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

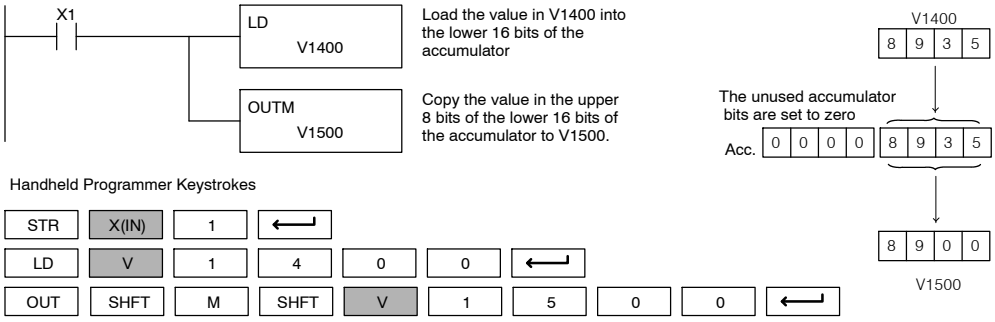
The Out Most instruction copies the value in the upper eight bits of the lower sixteen bits of the accumulator to the upper eight bits of the specified V-memory location (i.e., it copies the high byte of the low word of the accumulator).



Operand Data Type		DL450 Range
	A	**aaa**
V-memory	V	All (See p. 3-42)
Pointer	P	All (See p. 3-42)

In the following example, when X1 is on, the value in V1400 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the upper 8 bits of the lower 16 bits of the accumulator is copied to V1500 using the Out Most instruction.

DirectSOFT



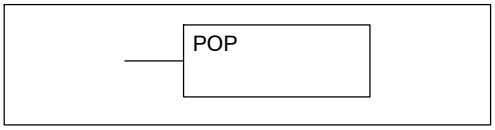
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
OUT	SHFT	M	SHFT	V	1	5	0	0	↵

Pop (POP)

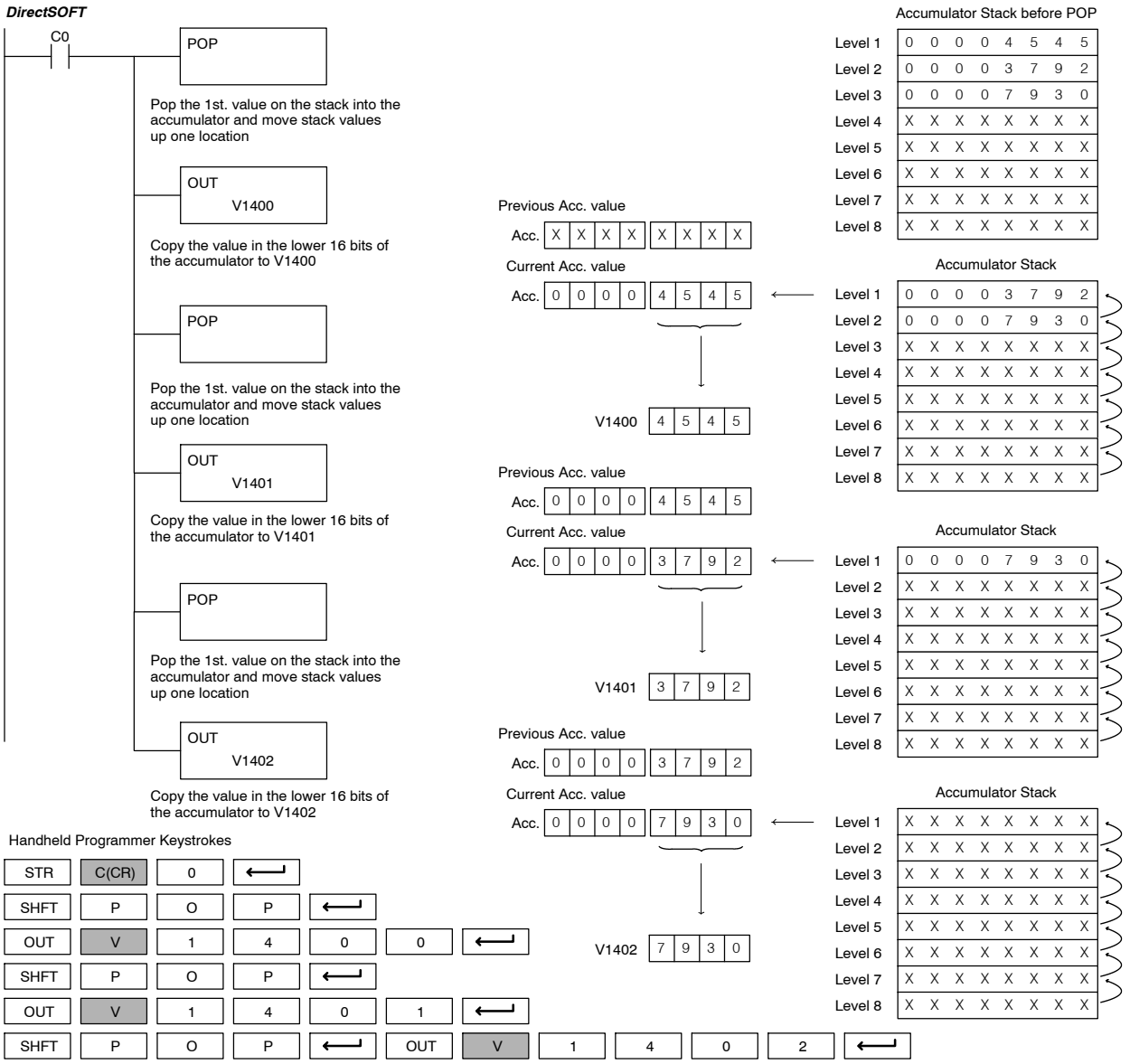
✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

The Pop instruction moves the value from the first level of the accumulator stack (32 bits) to the accumulator and shifts each value in the stack up one level.



In the example below, when C0 is on the Pop instruction moves the value 4545 currently on top of the stack into the accumulator. The value is output to V1400 using the Out instruction. The next Pop moves the value 3792 into the accumulator and outputs the value to V1401. The last Pop moves the value 7930 into the accumulator and outputs it to V1402. Remember to use Out Double instructions if the value in the stack uses more than 16 bits (4 digits). Each value will occupy two V-memory locations.

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

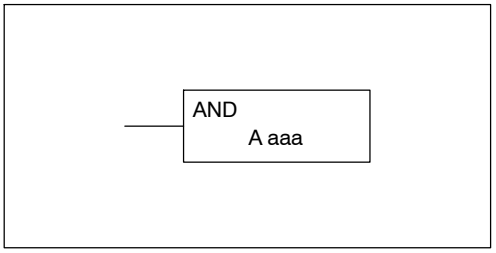


Accumulator Logic Instructions

And (AND)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

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



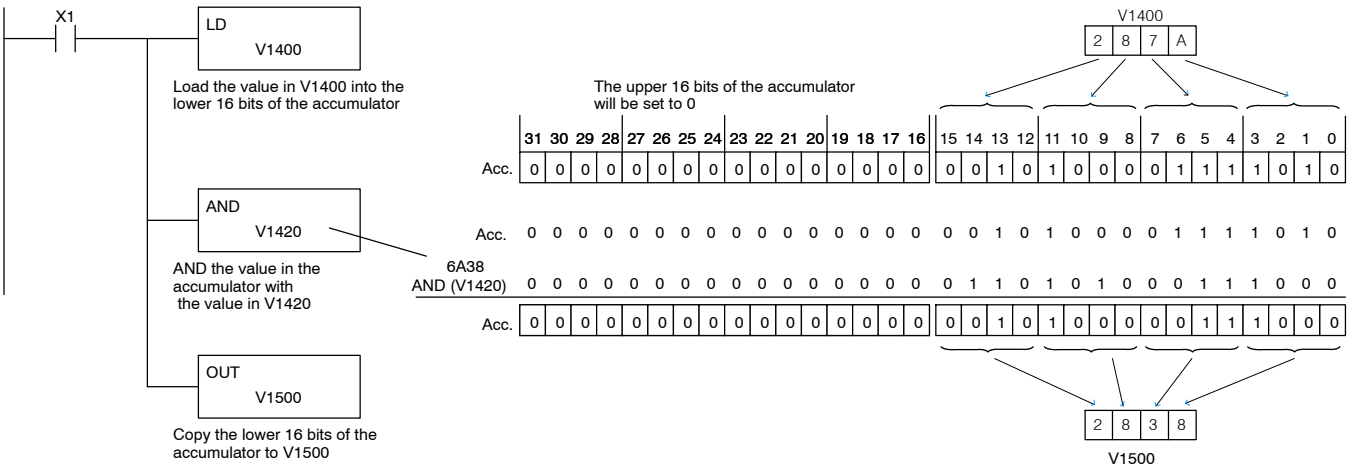
Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	--	All (See p. 3-41)	All (See p. 3-42)

Discrete Bit Flags	Description
SP63	Will be 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 V1400 will be loaded into the accumulator using the Load instruction. The value in the accumulator is Anted with the value in V1420 using the And instruction. The value in the lower 16 bits of the accumulator is output to V1500 using the Out instruction.

DirectSOFT



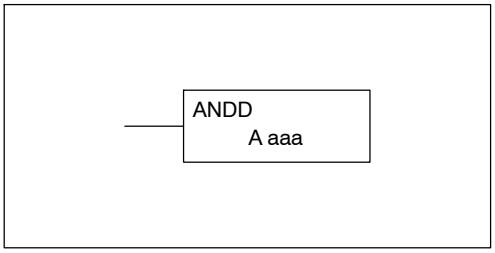
Handheld Programmer Keystrokes

STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
AND	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵

And Double (ANDD)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

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



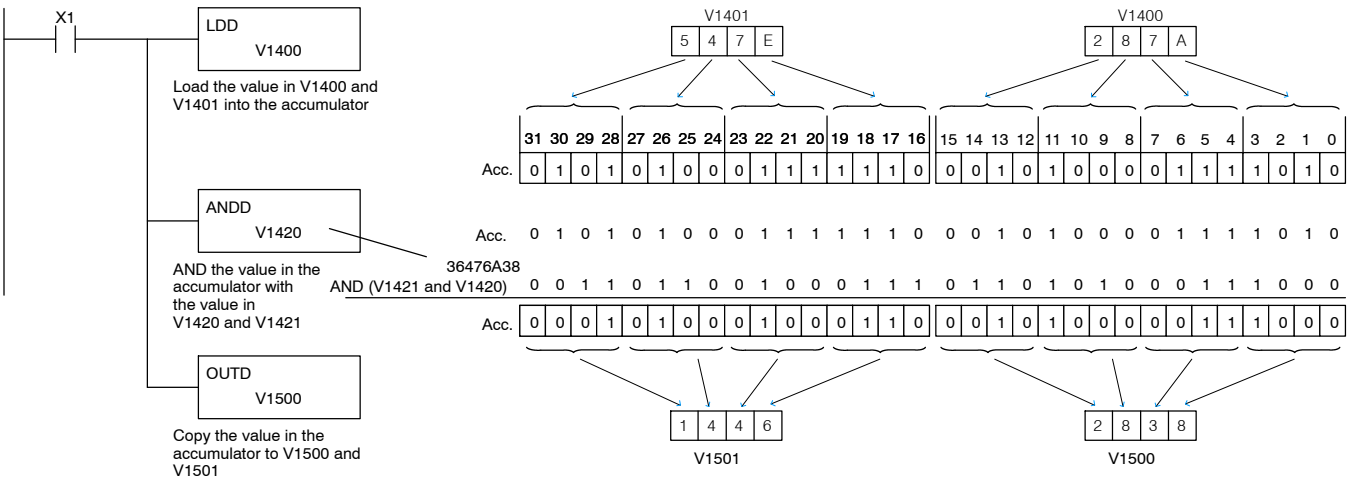
Operand Data Type		DL430 Range	DL440 Range	DL440 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	--	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	--	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-FFFF	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is 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 Load Double instruction. The value in the accumulator is Anted with V1420 and V1421 using the And double instruction. The value in the accumulator is output to V1500 and V1501 using the Out Double instruction.

DirectSOFT



Handheld Programmer Keystrokes

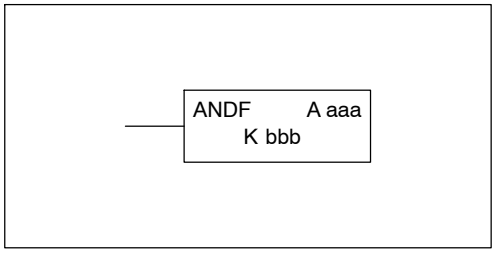
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
AND	SHFT	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

And Formatted (ANDF)

430	440	450
	✓	✓

DS	HPP
✓	✓

The And Formatted instruction logically ANDs the binary value in the accumulator and 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).



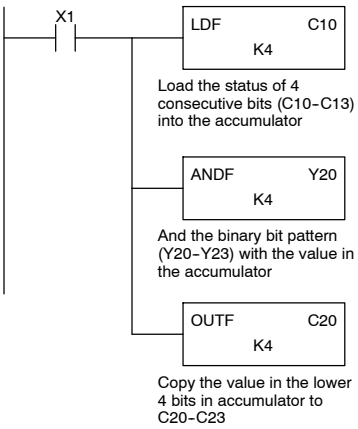
Operand Data Type		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb
Inputs	X	0-477	--	0-1777	--
Outputs	Y	0-477	--	0-1777	--
Control Relays	C	0-1777	--	0-3777	--
Stage Bits	S	0-1777	--	0-1777	--
Timer Bits	T	0-377	--	0-377	--
Counter Bits	CT	0-177	--	0-377	--
Special Relays	SP	0-137 320-717	--	0-137 320-717	--
Global I/O	GX	0-1777	--	0-2777	--
Constant	K	--	1-32	--	1-32

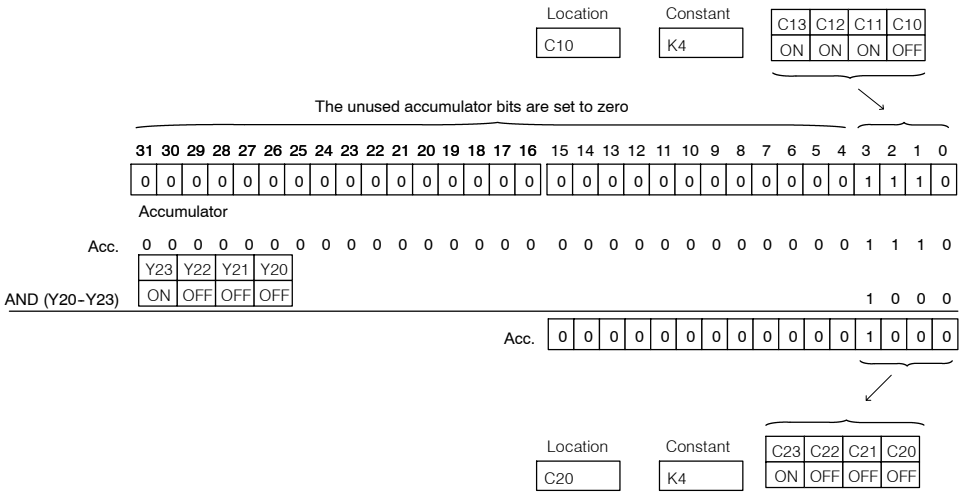
Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is the result in the accumulator is negative

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

In the following example, when X1 is on the Load Formatted instruction loads C10-C13 (4 binary bits) into the accumulator. The accumulator contents is logically ANDed with the bit pattern from Y20-Y23 using the And Formatted instruction. The Out Formatted instruction outputs the accumulator's lower four bits to C20-C23.

DirectSOFT





Handheld Programmer Keystrokes

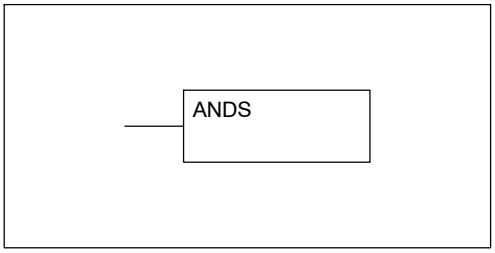
STR	X(IN)	1	↵						
LD	SHFT	F	SHFT	C(CR)	1	0	K(CON)	4	↵
AND	SHFT	F	SHFT	Y(OUT)	2	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	C(CR)	2	0	K(CON)	4	↵

And with Stack (ANDS)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

The And with Stack instruction is a 32 bit instruction that logically ands the value in the accumulator with the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed from the stack and all values are moved up one level. Discrete status flags indicate if the result of the And with Stack is zero or a negative number (the most significant bit is on).

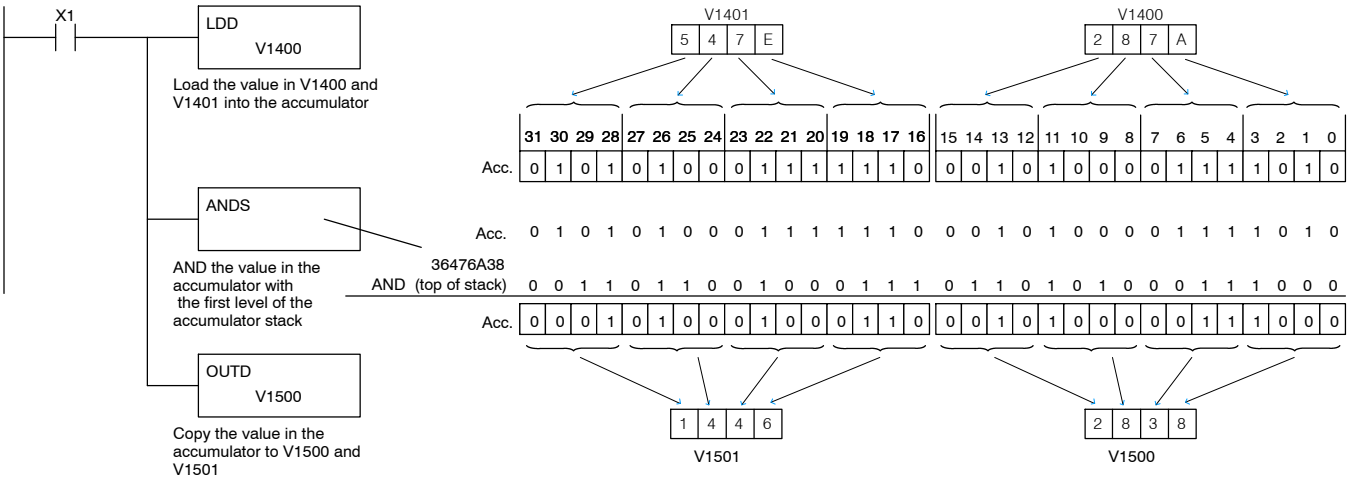


Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is the result in the accumulator is negative

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 Anted 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.

DirectSOFT



Handheld Programmer Keystrokes

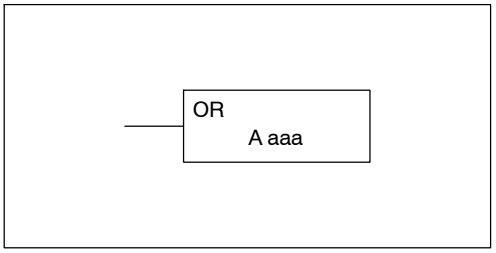
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
AND	SHFT	S	↵						
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Or (OR)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

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



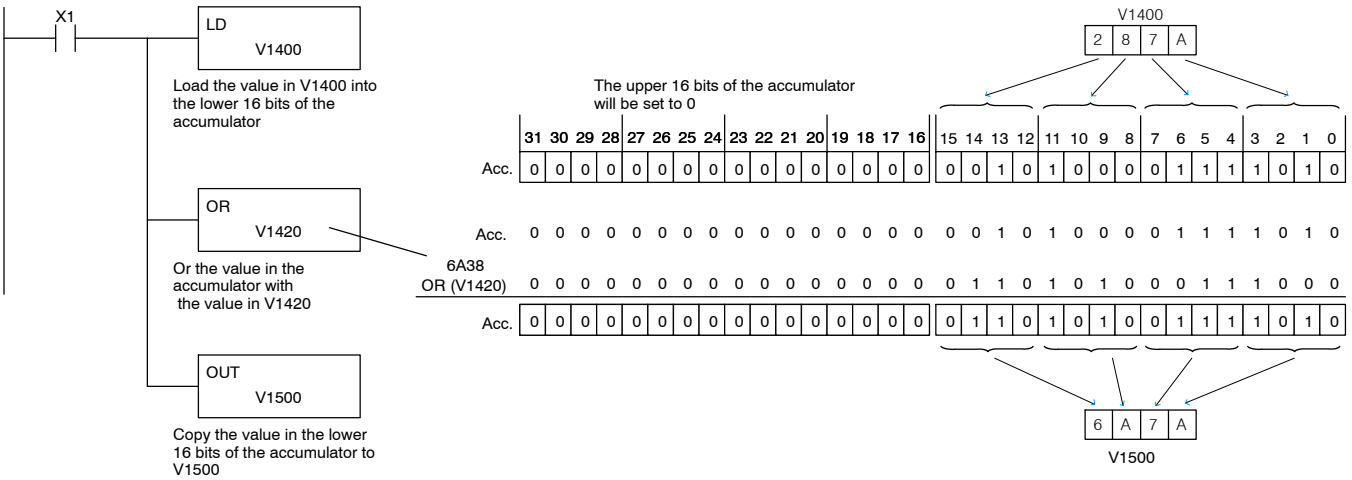
Operand Data Type		DL430 Range	DL440 Range	DL440 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	--	All V mem (See p. 3-41)	All V mem (See p. 3-42)

Discrete Bit Flags	Description
SP63	Will be 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 V1400 will be loaded into the accumulator using the Load instruction. The value in the accumulator is Hoped with V1420 using the Or instruction. The value in the lower 16 bits of the accumulator is output to V1500 using the Out instruction.

DirectSOFT



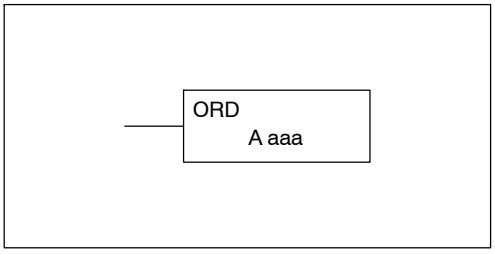
Handheld Programmer Keystrokes

STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
OR	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵

Or Double (ORD)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

The OR Double is a 32 bit instruction that 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 Or Double is zero or a negative number (the most significant bit is on).



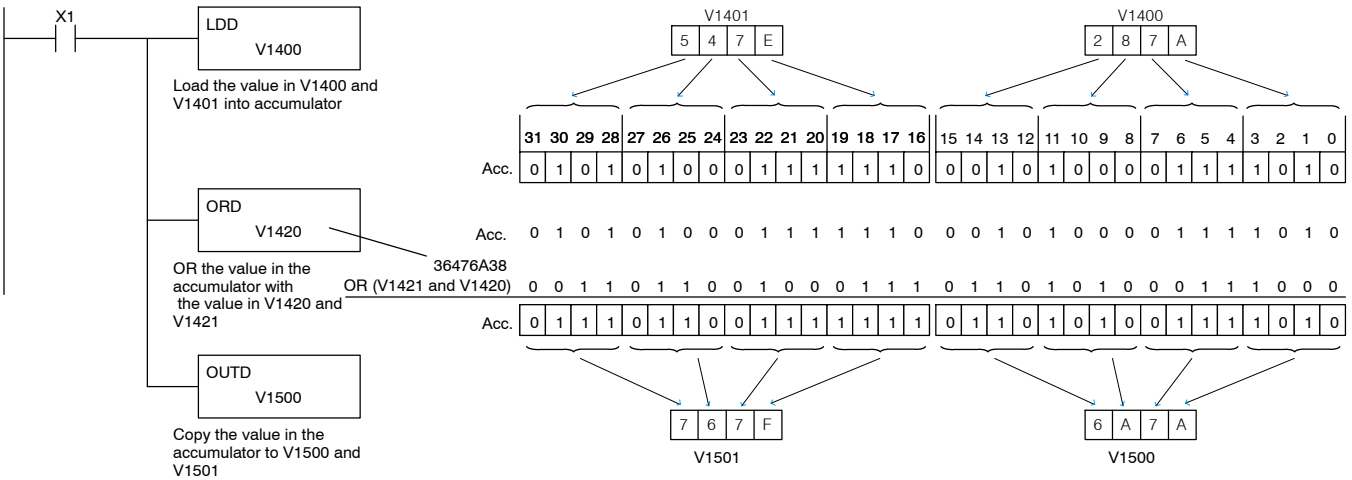
Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	--	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	--	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-FFFF	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is 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 Load Double instruction. The value in the accumulator is Hoped with V1420 and V1421 using the Or Double instruction. The value in the accumulator is output to V1500 and V1501 using the Out Double instruction.

DirectSOFT



Handheld Programmer Keystrokes

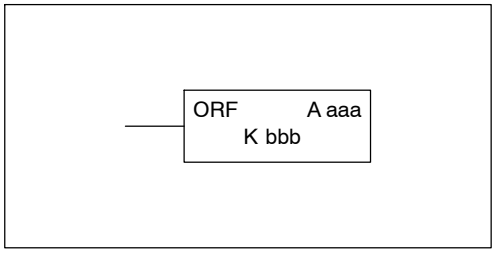
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
OR	SHFT	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Or Formatted (ORF)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

The Or Formatted 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).



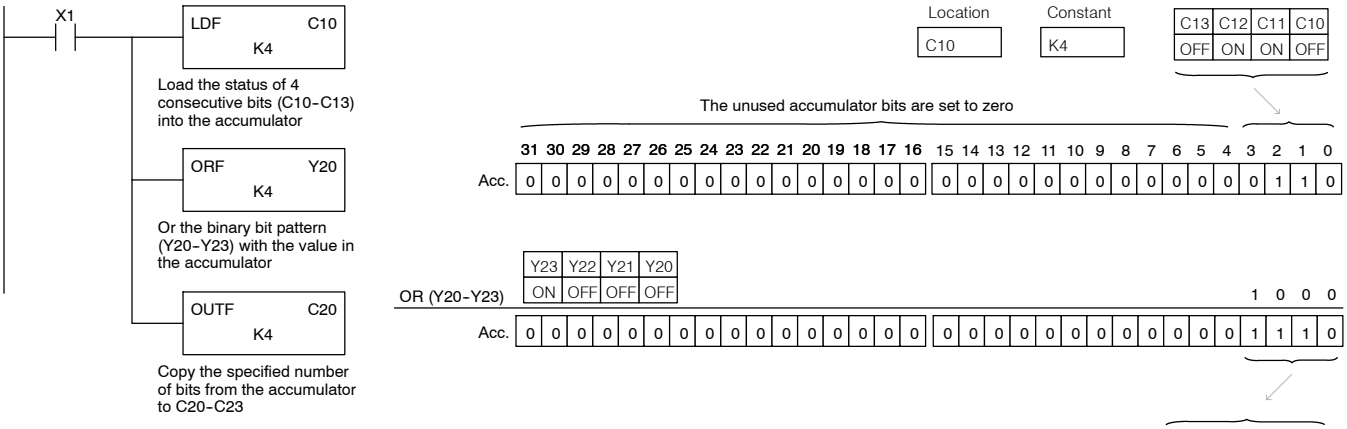
Operand Data Type		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb
Inputs	X	0-477	--	0-1777	--
Outputs	Y	0-477	--	0-1777	--
Control Relays	C	0-1777	--	0-3777	--
Stage Bits	S	0-1777	--	0-1777	--
Timer Bits	T	0-377	--	0-377	--
Counter Bits	CT	0-177	--	0-377	--
Special Relays	SP	0-137 320-717	--	0-137 320-717	--
Global I/O	GX	0-1777	--	0-2777	--
Constant	K	--	1-32	--	1-32

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is the result in the accumulator is negative

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

In the following example, when X1 is on the Load Formatted instruction loads C10-C13 (4 binary bits) into the accumulator. The Or Formatted instruction logically ORs the accumulator contents with Y20-Y23 bit pattern. The Out Formatted instruction outputs the accumulator's lower four bits to C20-C23.

DirectSOFT



Handheld Programmer Keystrokes

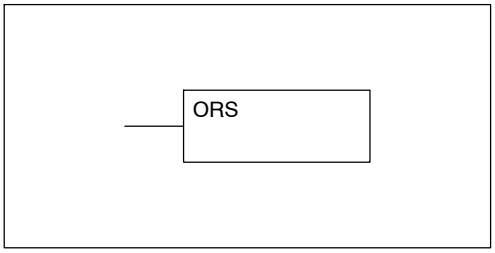
STR	X(IN)	1	↵						
LD	SHFT	F	SHFT	C(CR)	1	0	K(CON)	4	↵
OR	SHFT	F	SHFT	Y(OUT)	2	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	C(CR)	2	0	K(CON)	4	↵

Or with Stack (ORS)

430	440	450
	✓	✓

DS	HPP
✓	✓

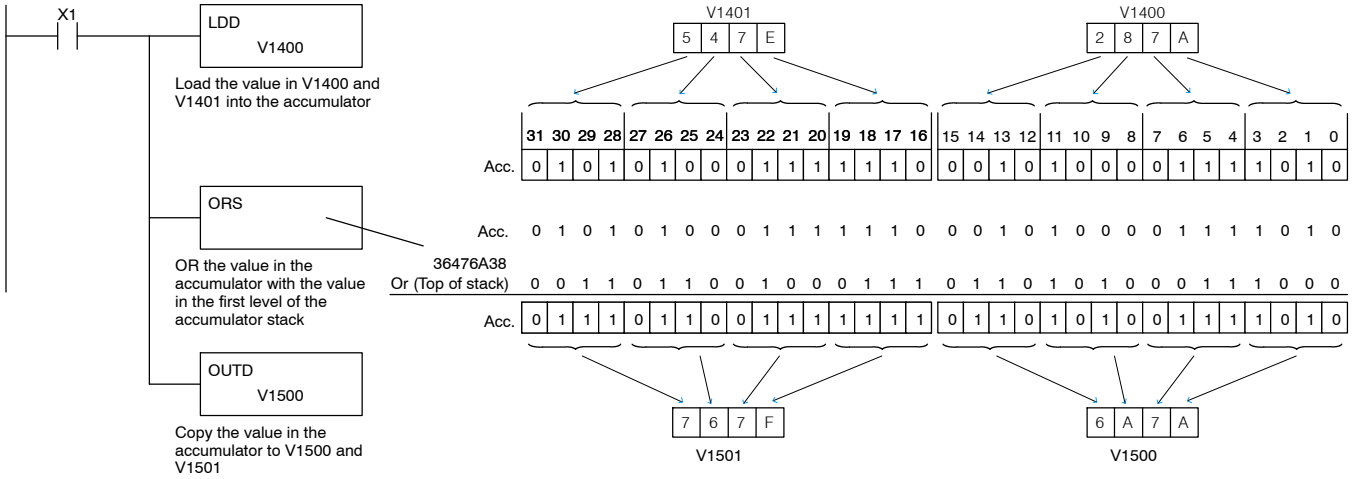
The Or with Stack instruction is a 32 bit instruction that logically ors the value in the accumulator with the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed from the stack and all values are moved up one level. Discrete status flags indicate if the result of the Or with Stack is zero or a negative number (the most significant bit is on).



Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is the result in the accumulator is negative

In the following example when X1 is on, the binary value in the accumulator will be Hoped with the binary value in the first level of the stack. The result resides in the accumulator.

DirectSOFT



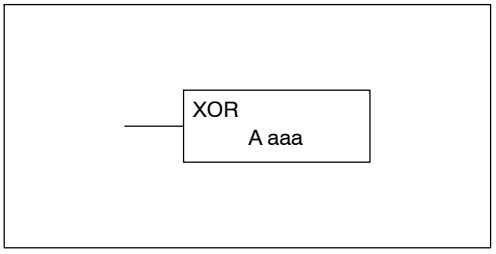
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
OR	SHFT	S	↵						
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Exclusive Or (XOR)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

The Exclusive Or instruction is a 16 bit instruction that performs an exclusive or of the value in the lower 16 bits of the accumulator and a specified V-memory location (Aaaa). The discrete status flag indicates if the result of the Xor is zero.



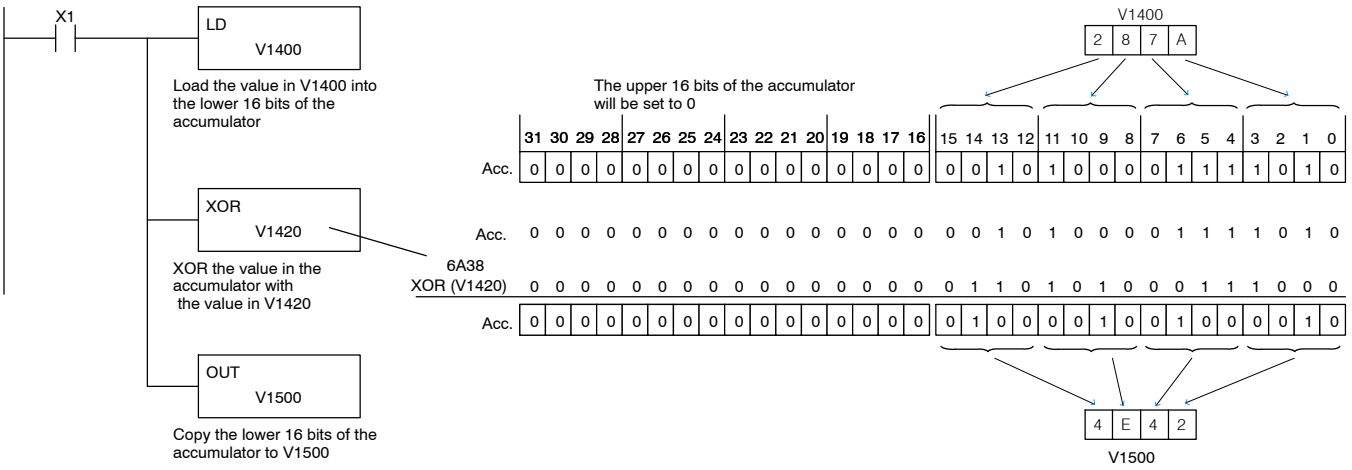
Operand Data Type		DL430 Range	DL440 Range	DL440 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	--	All V mem (See p. 3-41)	All V mem (See p. 3-42)

Discrete Bit Flags	Description
SP63	Will be 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 V1400 will be loaded into the accumulator using the Load instruction. The value in the accumulator is exclusive Hoped with V1420 using the Exclusive Or instruction. The value in the lower 16 bits of the accumulator is output to V1500 using the Out instruction.

DirectSOFT



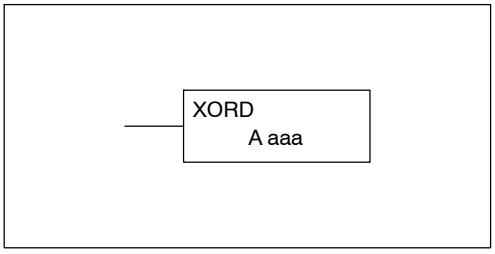
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
SHFT	X	SHFT	OR	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵			

Exclusive Or Double (XORD)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

The Exclusive OR Double 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 Exclusive Or Double is zero or a negative number (the most significant bit is on).

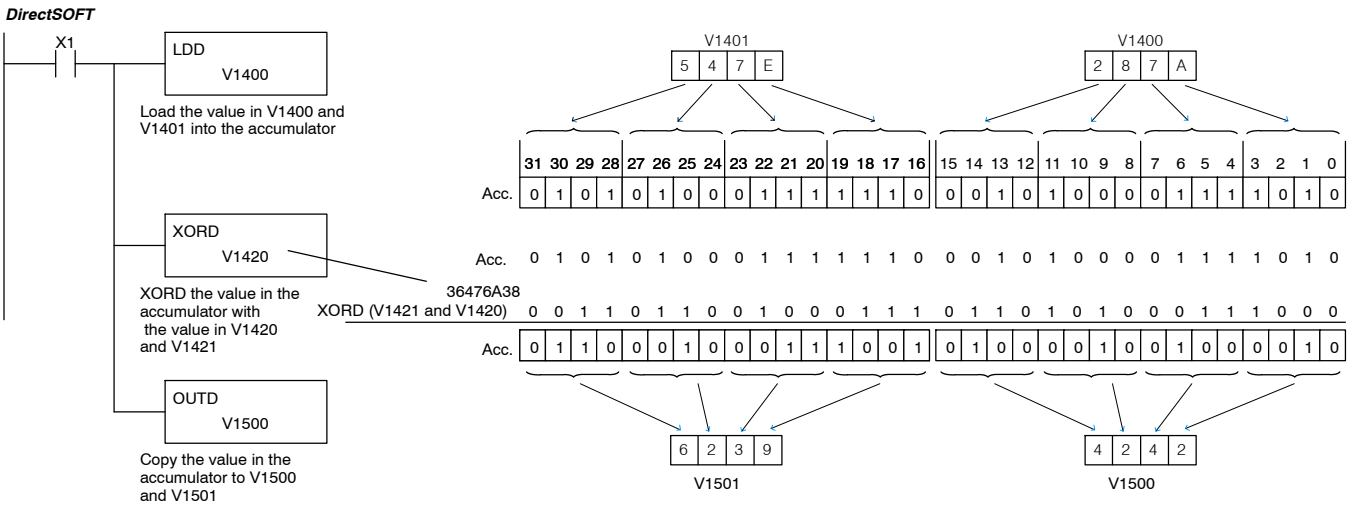


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	--	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	--	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-FFFF	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is 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 Load Double instruction. The value in the accumulator is exclusively Hoped with V1420 and V1421 using the Exclusive Or Double instruction. The value in the accumulator is output to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

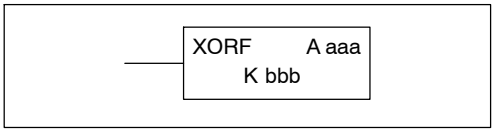
STR	X(IN)	1	↵									
LD	SHFT	D	SHFT	V	1	4	0	0	↵			
SHFT	X	SHFT	OR	SHFT	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵			

Exclusive Or Formatted (XORF)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

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



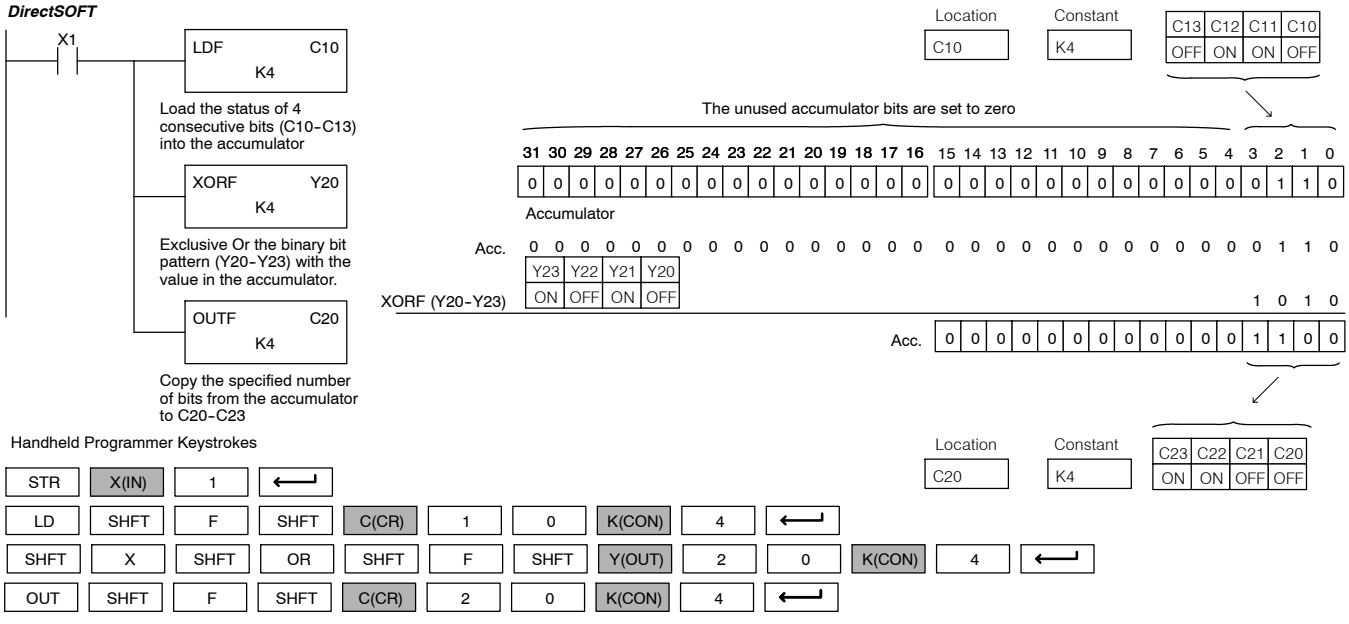
The instruction requires a starting location (Aaaa) and the number of bits (Bbbb) to be exclusive ORed. Discrete status flags indicate if the result of the Exclusive Or Formatted is zero or negative (the most significant bit =1).

Operand Data Type		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb
Inputs	X	0-477	--	0-1777	--
Outputs	Y	0-477	--	0-1777	--
Control Relays	C	0-1777	--	0-3777	--
Stage Bits	S	0-1777	--	0-1777	--
Timer Bits	T	0-377	--	0-377	--
Counter Bits	CT	0-177	--	0-377	--
Special Relays	SP	0-137 320-717	--	0-137 320-717	--
Global I/O	GX	0-1777	--	0-2777	--
Constant	K	--	1-32	--	1-32

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is the result in the accumulator is negative

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

In the following example, when X1 is on, the binary pattern of C10-C13 (4 bits) will be loaded into the accumulator using the Load Formatted instruction. The value in the accumulator will be logically Exclusive Hoped with the bit pattern from Y20-Y23 using the Exclusive Or Formatted instruction. The value in the lower 4 bits of the accumulator is output to C20-C23 using the Out Formatted instruction.

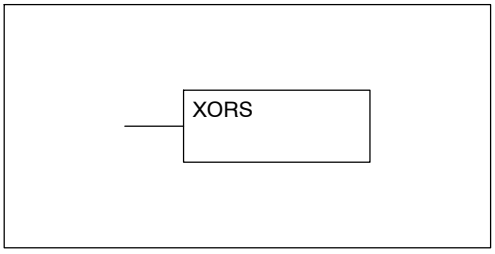


Exclusive Or with Stack (XORS)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

The Exclusive Or with Stack instruction is a 32 bit instruction that performs an exclusive or of the value in the accumulator with the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed from the stack and all values are moved up one level. Discrete status flags indicate if the result of the Exclusive Or with Stack is zero or a negative number (the most significant bit is on).

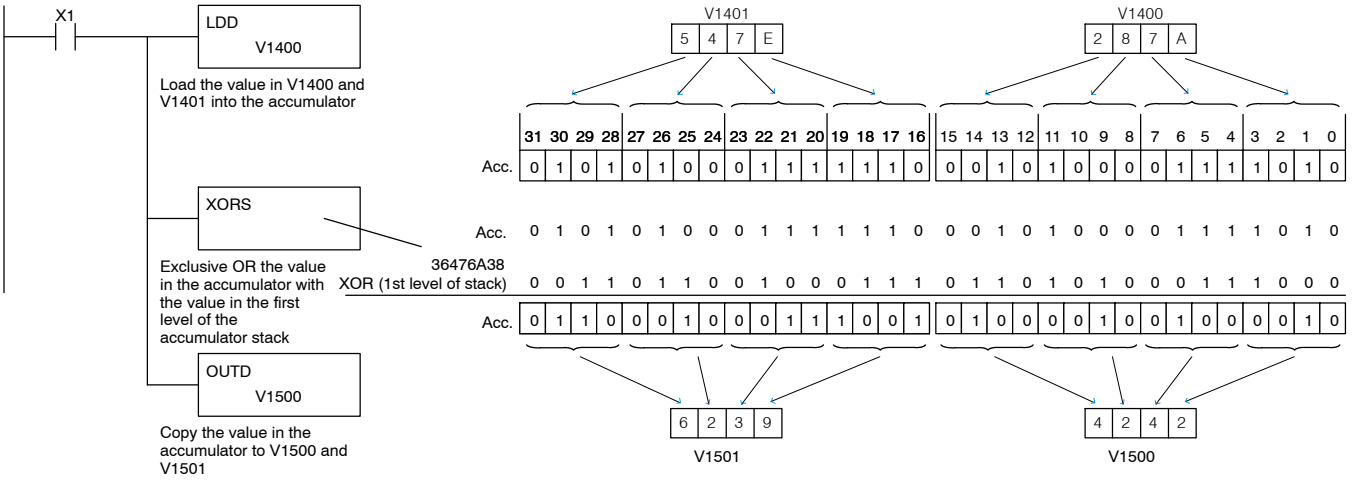


Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is the result in the accumulator is negative

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

In the following example when X1 is on, the binary value in the accumulator will be exclusive Hoped with the binary value in the first level of the accumulator stack. The result will reside in the accumulator.

DirectSOFT



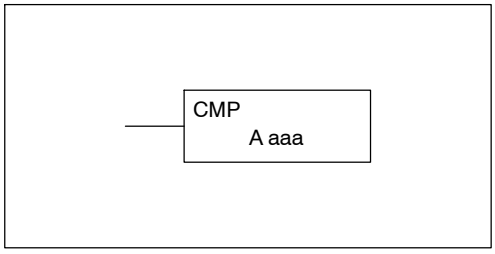
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	X	SHFT	OR	SHFT	S	↵			
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Compare (CMP)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

The compare instruction is a 16 bit instruction that compares the value in the lower 16 bits of the accumulator with the value in a specified V-memory location (Aaaa). The corresponding status flag will be turned on indicating the result of the comparison. You can compare either binary or BCD numbers, as long as both numbers are of the same data type.



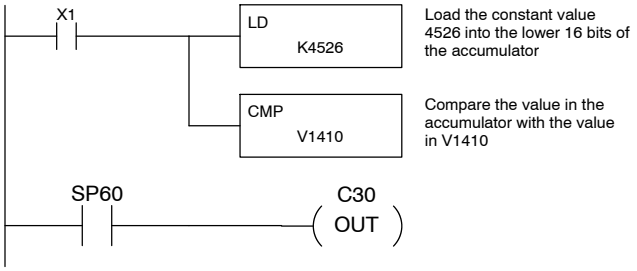
Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	--	All (See p. 3-41)	All (See p. 3-42)

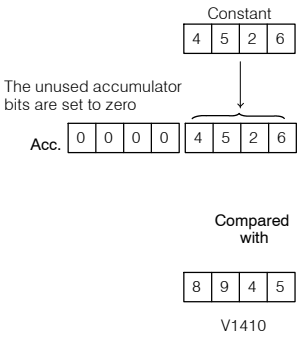
Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.

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

In the following example when X1 is on, the constant 4526 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the accumulator is compared with the value in V1410 using the Compare instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on energizing C30.

DirectSOFT





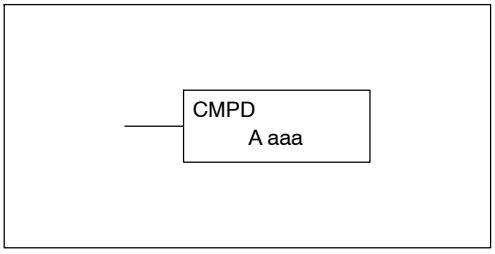
Handheld Programmer Keystrokes

STR	X(IN)	1	↵			
LD	K(CON)	4	5	2	6	↵
CMP	V	1	4	1	0	↵
STR	SPCL	6	0	↵		
OUT	C(CR)	3	0	↵		

Compare Double (CMPD)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

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. You can compare either binary or BCD numbers, as long as both numbers are of the same data type.



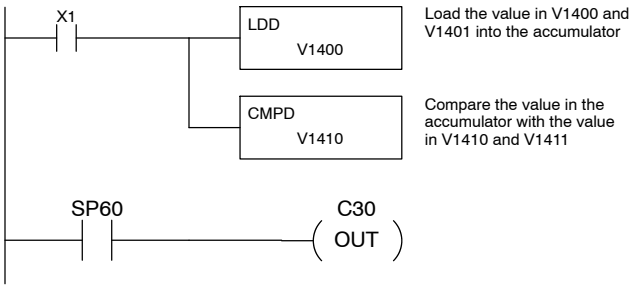
Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	--	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	--	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-FFFFFFFF	0-FFFFFFFF	0-FFFFFFFF

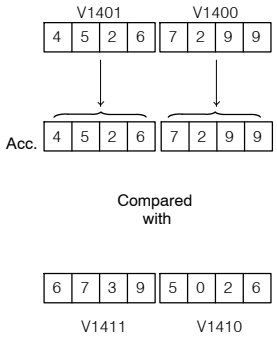
Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.

NOTE: 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 compared with the value in V1410 and V1411 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.

DirectSOFT





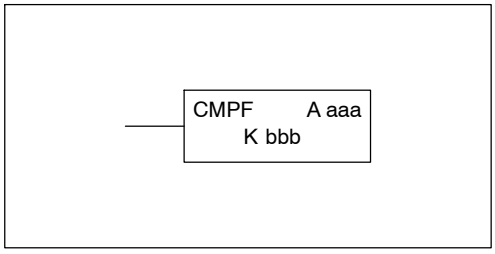
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
CMP	SHFT	D	SHFT	V	1	4	1	0	↵
STR	SPCL	6	0	↵					
OUT	C(CR)	3	0	↵					

Compare Formatted (CMPF)

430	440	450	DS	HPP
✗	✓	✓	✓	✓

The Compare Formatted compares the value in the accumulator with a specified number of discrete locations (1–32). The instruction requires a starting location (Aaaa) and the number of bits (Kbbb) to be compared. The corresponding status flag will be turned on indicating the result of the comparison.



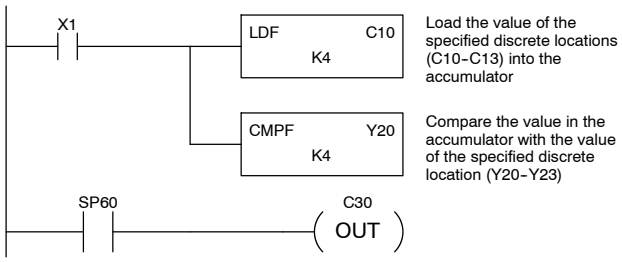
Operand Data Type		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb
Inputs	X	0–477	--	0–1777	--
Outputs	Y	0–477	--	0–1777	--
Control Relays	C	0–1777	--	0–3777	--
Stage Bits	S	0–1777	--	0–1777	--
Timer Bits	T	0–377	--	0–377	--
Counter Bits	CT	0–177	--	0–377	--
Special Relays	SP	0–137 320–717	--	0–137 320–717	--
Global I/O	GX	0–1777	--	0–2777	--
Constant	K	--	1–32	--	1–32

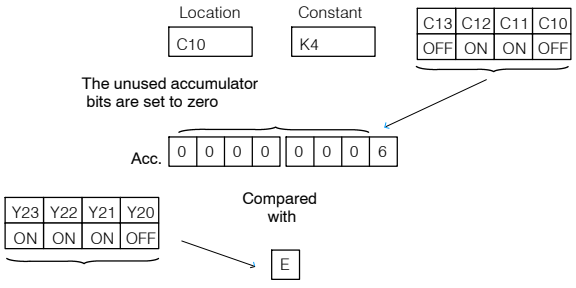
Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.

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

In the following example, when X1 is on the Load Formatted instruction loads the binary value (6) from C10–C13 into the accumulator. The CMPF instruction compares the value in the accumulator to the value in Y20–Y23 (E hex). The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on energizing C30.

DirectSOFT





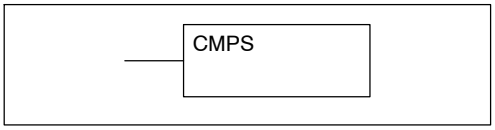
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	F	SHFT	C	1	0	K(CON)	4	
CMP	SHFT	F	SHFT	Y(OUT)	2	0	K(CON)	4	↵
STR	SPCL	6	0	↵					
OUT	C(CR)	3	0	↵					

Compare with Stack (CMPS)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

The Compare with Stack instruction is a 32-bit instruction that compares the value in the accumulator with the value in the first level of the accumulator stack.



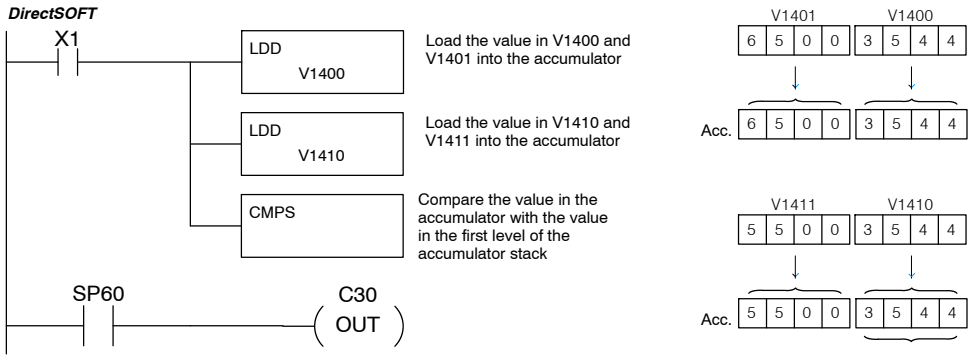
The corresponding status flag will be turned on indicating the result of the comparison. This does not affect the value in the accumulator.

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.

See Errata Sheet at the beginning of this file. The descriptions in this table have been updated.

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

In the following example when X1 is on, the value in V1400 and V1401 is loaded into the accumulator using the Load Double instruction. The value in V1410 and V1411 is loaded into the accumulator using the Load Double instruction. The value that was loaded into the accumulator from V1400 and V1401 is placed on top of the stack when the second Load instruction is executed. The value in the accumulator is compared with the value in the first level or the accumulator stack using the CMPS instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value in the stack, SP60 will turn on, energizing C30.



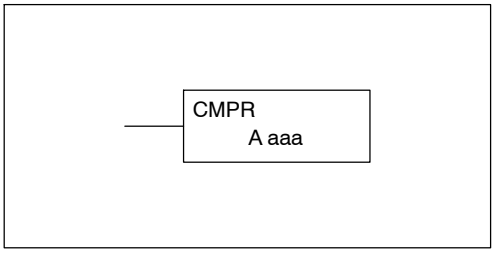
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
LD	SHFT	D	SHFT	V	1	4	1	0	↵
CMP	SHFT	S	↵						
STR	SPCL	6	0	↵					
OUT	C(CR)	3	0	↵					

Compare Real Number (CMPR)

430	440	450
		✓
✓		✓
DS		HPP

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.



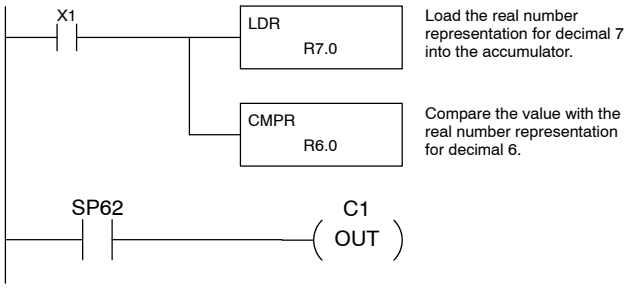
Operand Data Type		DL450 Range
	A	aaa
V-memory	V	All (See p. 3-42)
Pointer	P	All (See p. 3-42)
Constant	R	-3.402823E+038 to + -3.402823E+038

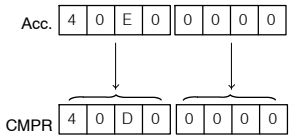
Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.
SP71	On anytime the V-memory specified by a pointer (P) is not valid.
SP75	On when a real number instruction is executed and a non-real number was encountered.

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

In the following example when X1 is on, the LDR instruction loads the real number representation for 7 decimal into the accumulator. The CMPR instruction compares the accumulator contents with the real representation for decimal 6. Since 7 > 6, the corresponding discrete status flag is turned on (special relay SP62).

DirectSOFT





Handheld Programmer Keystrokes

STR	X(IN)	1	LD	SHFT	D	SHFT	↵	
K(CON)	4	0	E	0	0	0	0	0 ↵
CMP	SHFT	R	SHFT					
K(CON)	4	0	D	0	0	0	0	0 ↵
STR	SPCL	6	0	↵				
OUT	C(CR)	3	0	↵				

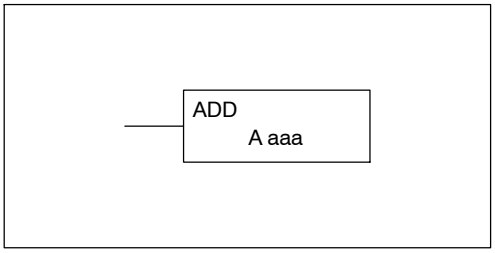
Math Instructions

Add (ADD)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

Add is a 16 bit instruction that adds a BCD value in the accumulator with a BCD value in a V-memory location (Aaaa). The result resides in the accumulator.

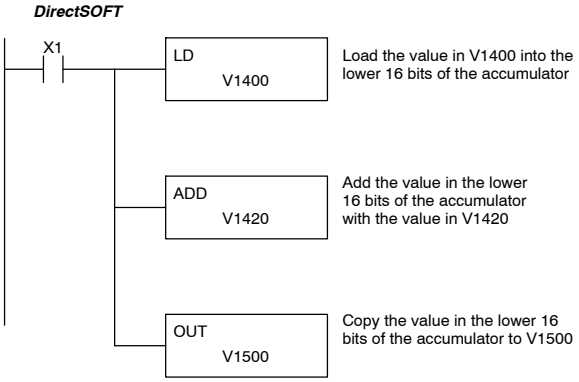


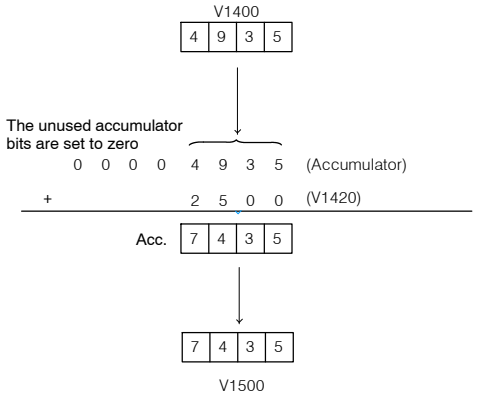
Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)

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 V1400 will be loaded into the accumulator using the Load instruction. The value in the lower 16 bits of the accumulator are added to the value in V1420 using the Add instruction. The value in the accumulator is copied to V1500 using the Out instruction.





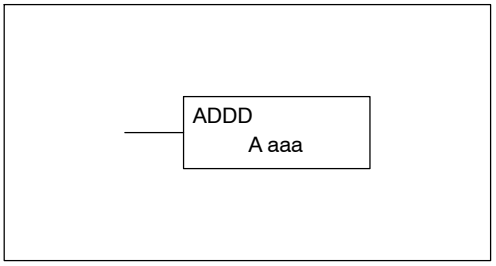
Handheld Programmer Keystrokes

STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
ADD	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵

Add Double (ADDD)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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.



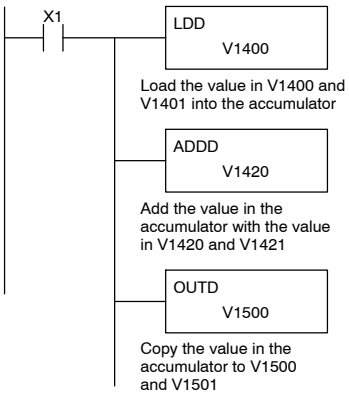
Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-99999999	0-99999999	0-99999999

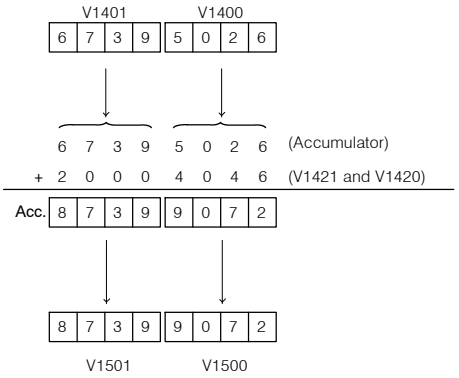
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16 bit addition instruction results in a carry.
SP67	On when the 32 bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number 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 added with the value in V1420 and V1421 using the Add Double instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

DirectSOFT





Handheld Programmer Keystrokes

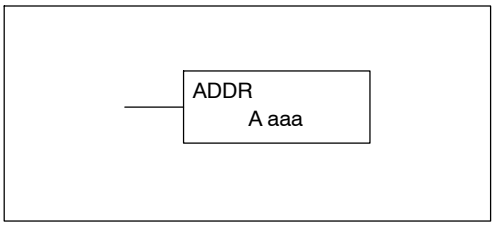
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
ADD	SHFT	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Add Real (ADDR)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

Add Real is a 32-bit instruction that adds a real number, which is either two consecutive V-memory locations or a 32-bit constant, to a real number in the accumulator. Both numbers must conform to the IEEE floating point format. The result is a 32-bit real number that resides in the accumulator.

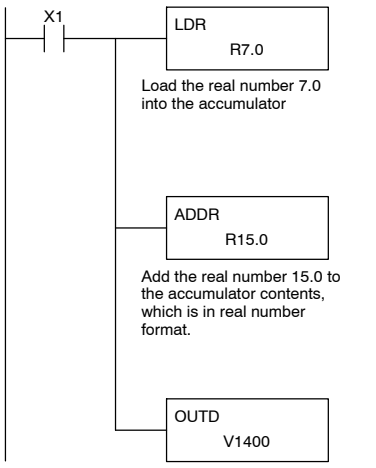


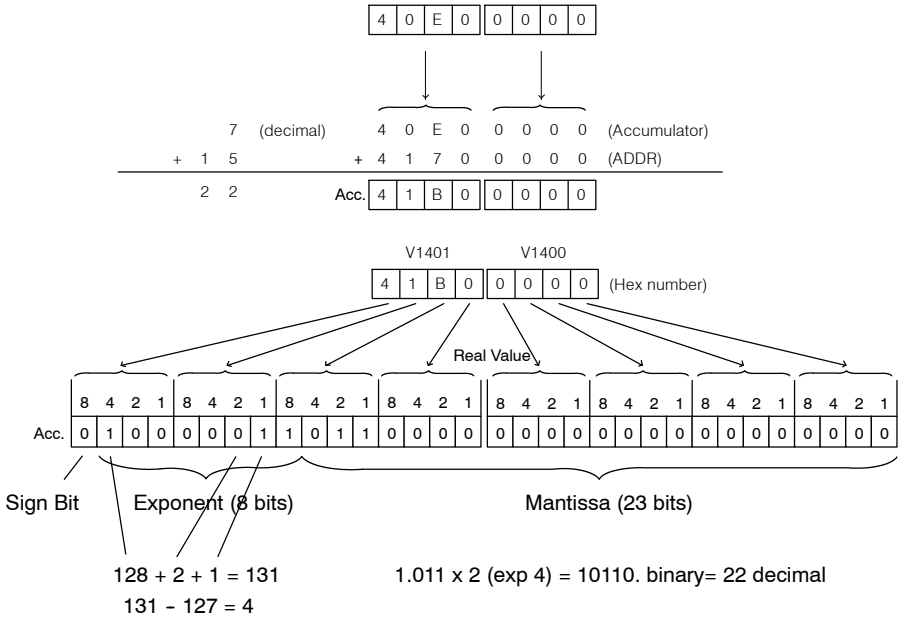
Operand Data Type		DL450 Range
	A	aaa
V-memory	V	All (See p. 3-42)
Pointer	P	All V mem (See p. 3-42)
Constant	R	-3.402823E+038 to +3.402823E+038

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP71	On anytime the V-memory specified by a pointer (P) is not valid.
SP72	On anytime the value in the accumulator is a valid floating point number.
SP73	on when a signed addition or subtraction results in a incorrect sign bit.
SP74	On anytime a floating point math operation results in an underflow error.
SP75	On when a real number instruction is executed and a non-real number was encountered.

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

DirectSOFT





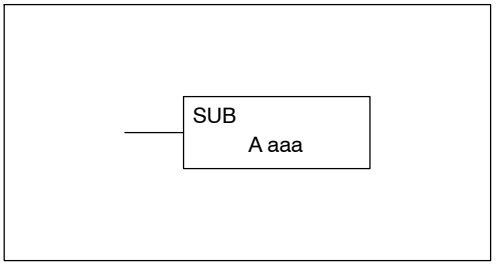
NOTE: If the value being added to a real number is 16,777,216 times smaller than the real number, the calculation will not work.

NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use ***Direct***SOFT for this feature.

Subtract (SUB)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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.

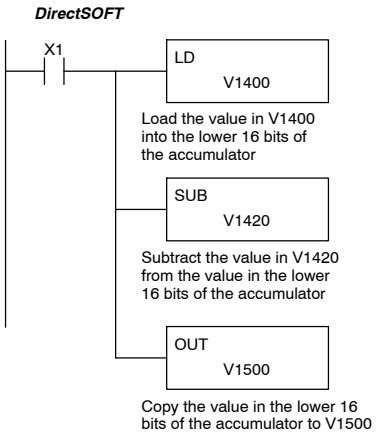


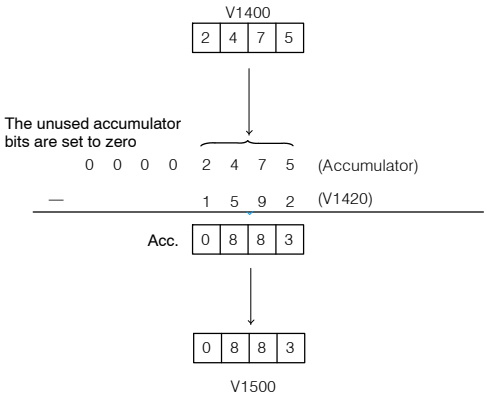
Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16 bit subtraction instruction results in a borrow.
SP65	On when the 32 bit subtraction instruction results in a borrow.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The value in V1420 is subtracted from the value in the accumulator using the Subtract instruction. The value in the accumulator is copied to V1500 using the Out instruction.





Handheld Programmer Keystrokes

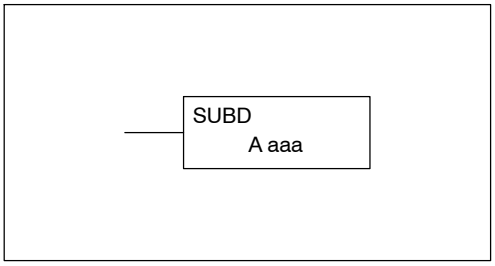
STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
SUB	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵

Subtract Double (SUBD)

✓ 430 ✓ 440 ✓ 450

✓ DS ✓ HPP

Subtract Double is a 32 bit instruction that subtracts the BCD value (Aaaa), which is either two consecutive V-memory locations or an 8-digit (max.) constant, from the BCD value in the accumulator. The result resides in the accumulator.

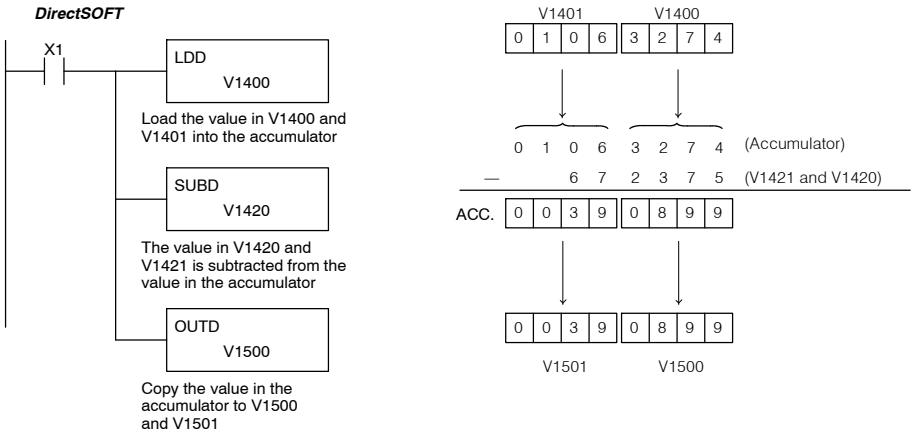


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-99999999	0-99999999	0-99999999

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16 bit subtraction instruction results in a borrow.
SP65	On when the 32 bit subtraction instruction results in a borrow.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

In the following example, when 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 subtracted from the value in the accumulator. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

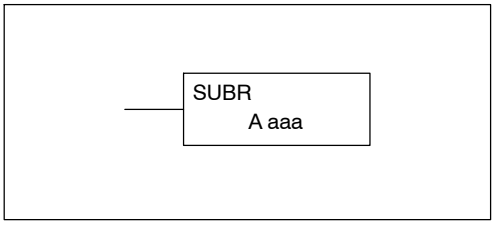
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SUB	SHFT	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Subtract Real (SUBR)

430	440	450
✓	✗	✗

DS	HPP
✓	✗

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. Both numbers must conform to the IEEE floating point format. The result is a 32-bit real number that resides in the accumulator.

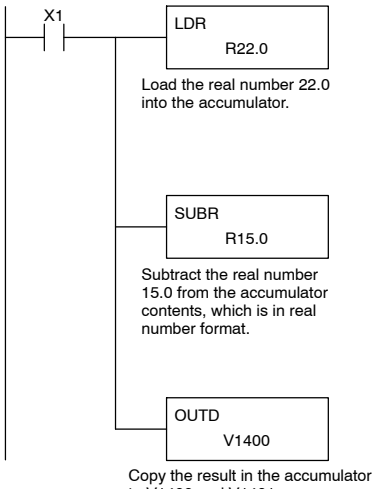


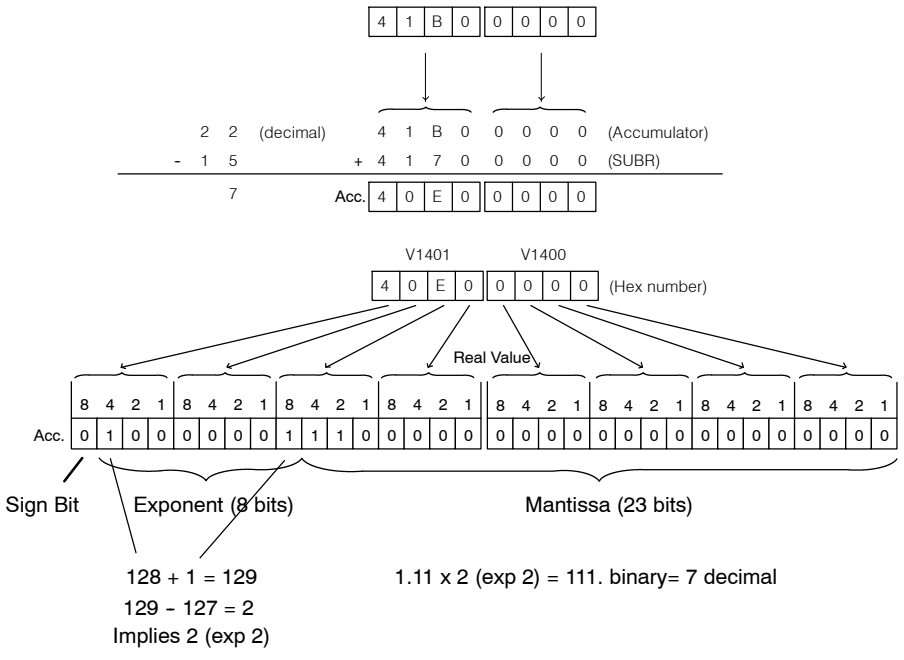
Operand Data Type		DL450 Range
	A	**aaa**
V-memory	V	All (See p. 3-42)
Pointer	P	All V mem (See p. 3-42)
Constant	R	-3.402823E+038 to +3.402823E+038

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP71	On anytime the V-memory specified by a pointer (P) is not valid.
SP72	On anytime the value in the accumulator is a valid floating point number.
SP73	on when a signed addition or subtraction results in a incorrect sign bit.
SP74	On anytime a floating point math operation results in an underflow error.
SP75	On when a real number instruction is executed and a non-real number was encountered.

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

DirectSOFT





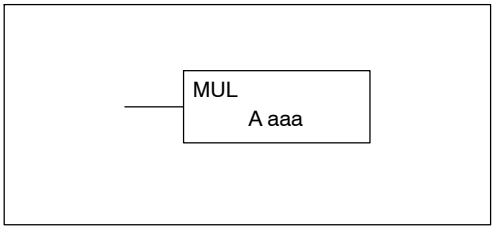
NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use ***Direct***SOFT for this feature.

Multiply (MUL)

✓ 430 ✓ 440 ✓ 450

✓ DS ✓ HPP

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.

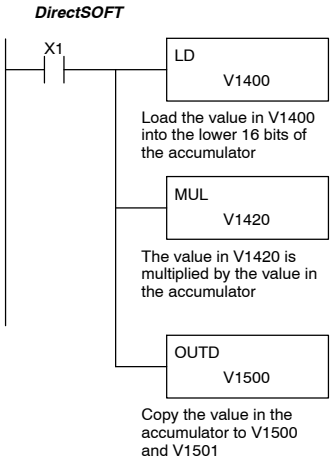


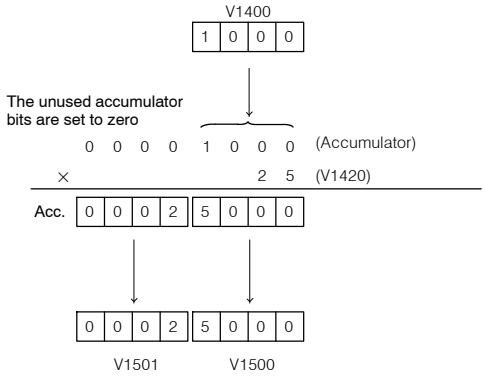
Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-9999	0-9999	0-9999

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number 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 multiplied by the value in the accumulator. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.





Handheld Programmer Keystrokes

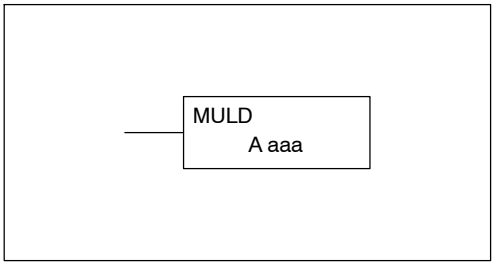
STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
MUL	V	1	4	2	0	↵			
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Multiply Double (MULD)

430	440	450
✗	✗	✓

DS	HPP
✓	✗

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.



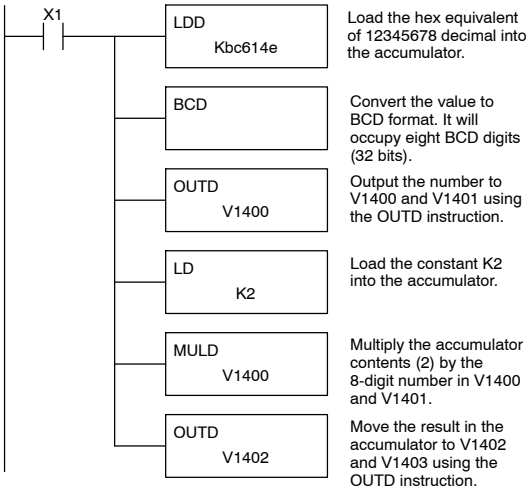
Operand Data Type		DL450 Range
	A	aaa
V-memory	V	All (See p. 3-42)
Pointer	P	--

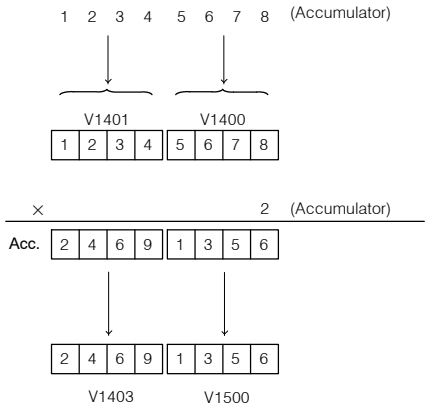
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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 is 24691356.

DirectSOFT





Handheld Programmer Keystrokes

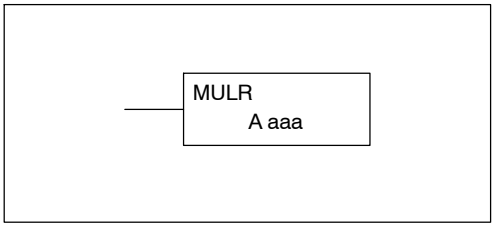
STR	X(IN)	1	↵	LD	SHFT	D	SHFT			
K(CON)	SHFT	B(H)	C(H)	6	1	4	E(H)	↵	BCD	↵
OUT	SHFT	D	SHFT	V	1	4	0	0	↵	
LD	K(CON)	2	↵	MUL	SHFT	D				
V	1	4	0	0	↵					
OUT	SHFT	D	SHFT	V	1	4	0	2	↵	

Multiply Real (MULR)

430	440	450
✓	✗	✗

DS	HPP
✓	✗

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 conform to the IEEE floating point format.

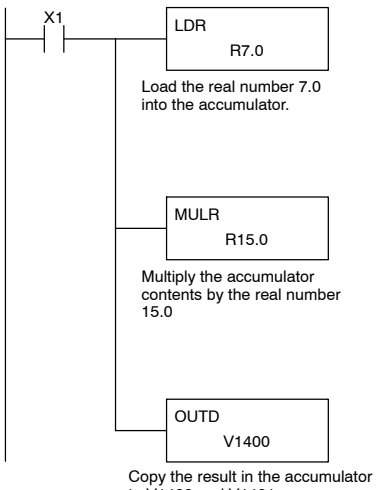


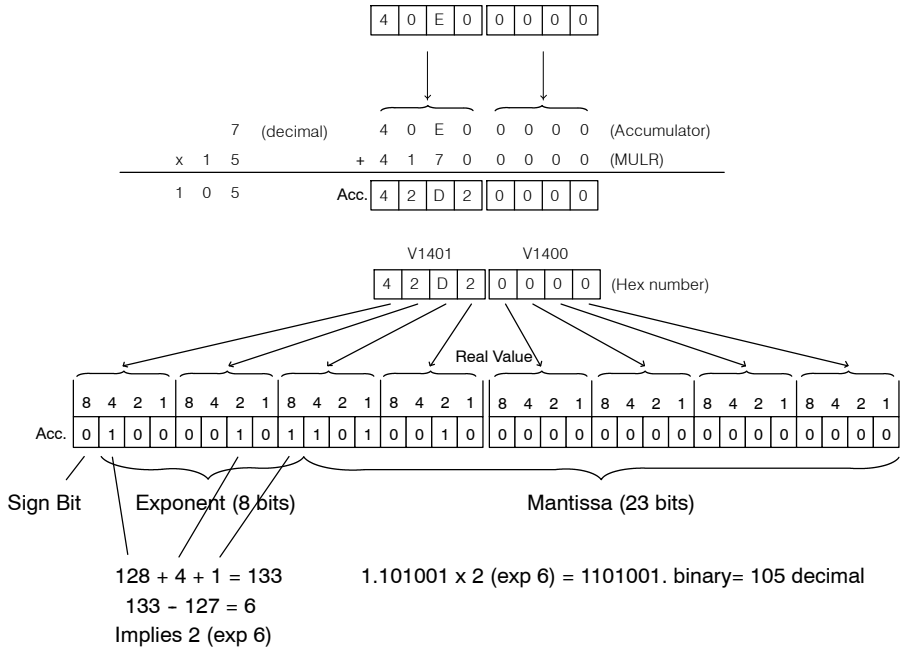
Operand Data Type		DL450 Range
	A	aaa
V-memory	V	All (See p. 3-42)
Pointer	P	All (See p. 3-42)
Constant	R	-3.402823E+038 to +3.402823E+038

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP71	On anytime the V-memory specified by a pointer (P) is not valid.
SP72	On anytime the value in the accumulator is a valid floating point number.
SP73	on when a signed addition or subtraction results in a incorrect sign bit.
SP74	On anytime a floating point math operation results in an underflow error.
SP75	On when a real number instruction is executed and a non-real number was encountered.

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

DirectSOFT



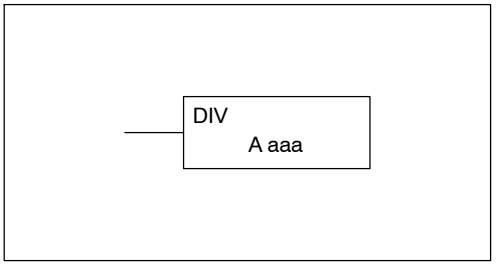


NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use ***Direct***SOFT for this feature.

Divide (DIV)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

Divide is a 16 bit instruction that divides the BCD value in the 32-bit 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.

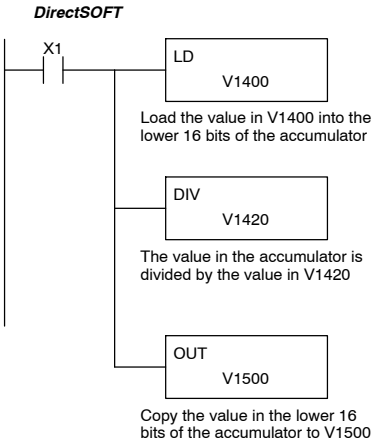


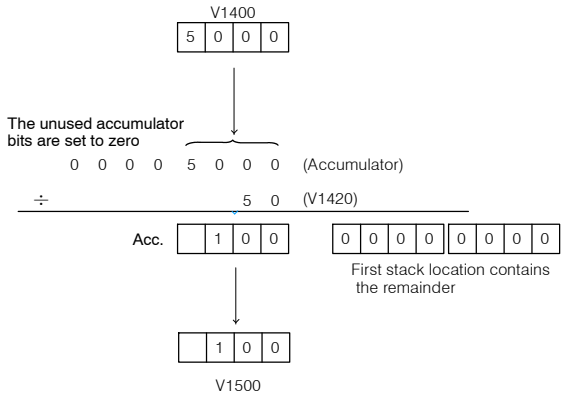
Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All V mem (See p. 3-42)
Constant	K	0-9999	0-9999	0-9999

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 V1400 will be loaded into the accumulator using the Load instruction. The value in the accumulator will be divided by the value in V1420 using the Divide instruction. The value in the accumulator is copied to V1500 using the Out instruction.





Handheld Programmer Keystrokes

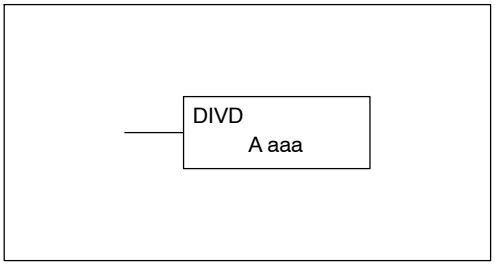
STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
DIV	V	1	4	2	0	↵
OUT	V	1	5	0	0	↵

Divide Double (DIVD)

430	440	450
	✓	✓

DS	HPP
✓	✓

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.

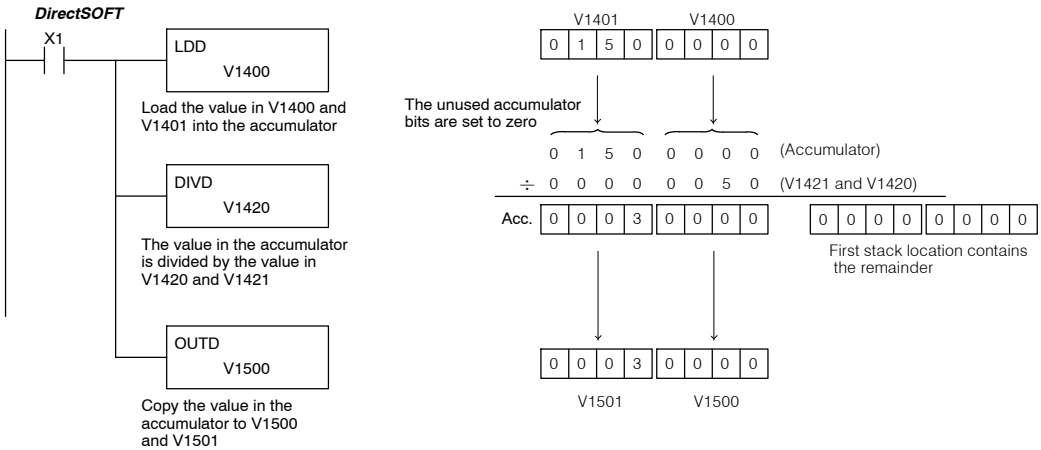


Operand Data Type		DL440 Range	DL450 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-41)	All (See p. 3-42)

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is divided by the value in V1420 and V1421 using the Divide Double instruction. The first part of the quotient resides in the accumulator and the remainder resides in the first stack location. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

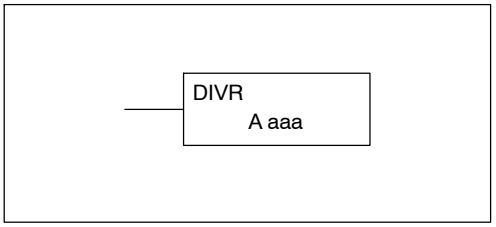
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
DIV	SHFT	D	SHFT	V	1	4	2	0	↵
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Divide Real (DIVR)

430	440	450
✓	✗	✗

DS	HPP
✓	✗

The Divide Real instruction divides a real number in the accumulator by either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must conform to the IEEE floating point format.

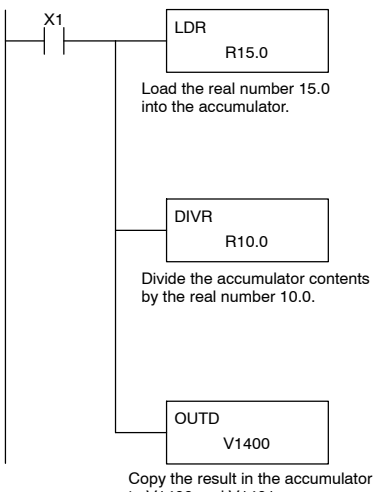


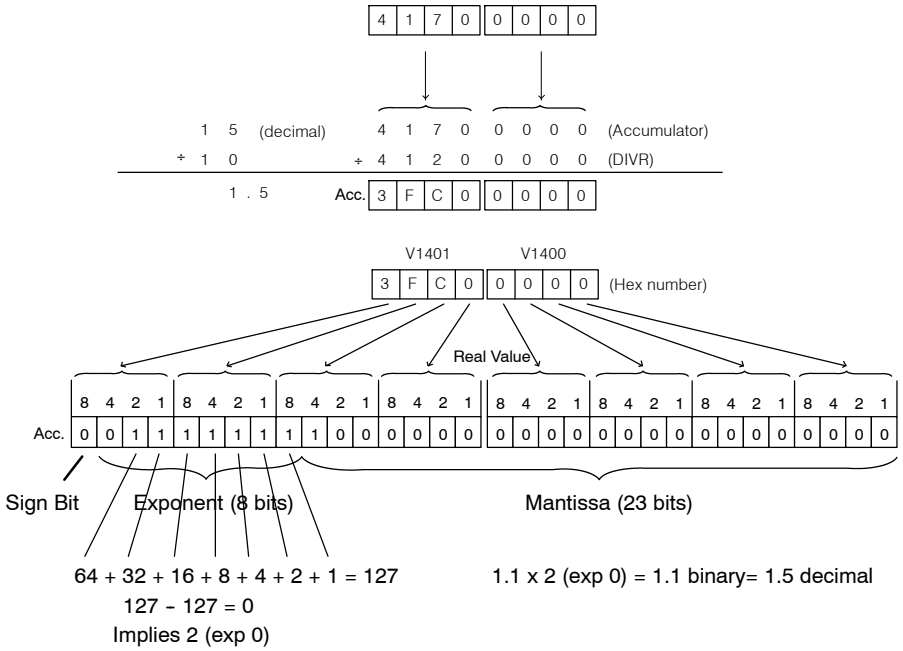
Operand Data Type		DL450 Range
	A	aaa
V-memory	V	All (See p. 3-42)
Pointer	P	All (See p. 3-42)
Constant	R	-3.402823E+038 to +3.402823E+038

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP71	On anytime the V-memory specified by a pointer (P) is not valid.
SP72	On anytime the value in the accumulator is a valid floating point number.
SP73	on when a signed addition or subtraction results in a incorrect sign bit.
SP74	On anytime a floating point math operation results in an underflow error.
SP75	On when a real number instruction is executed and a non-real number was encountered.

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

DirectSOFT





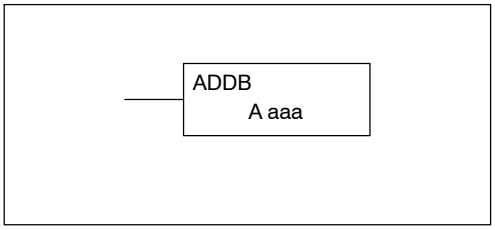
NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use ***Direct***SOFT for this feature.

Add Binary (ADDB)

✓ 430 ✓ 440 ✓ 450

✓ DS ✓ HPP

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.

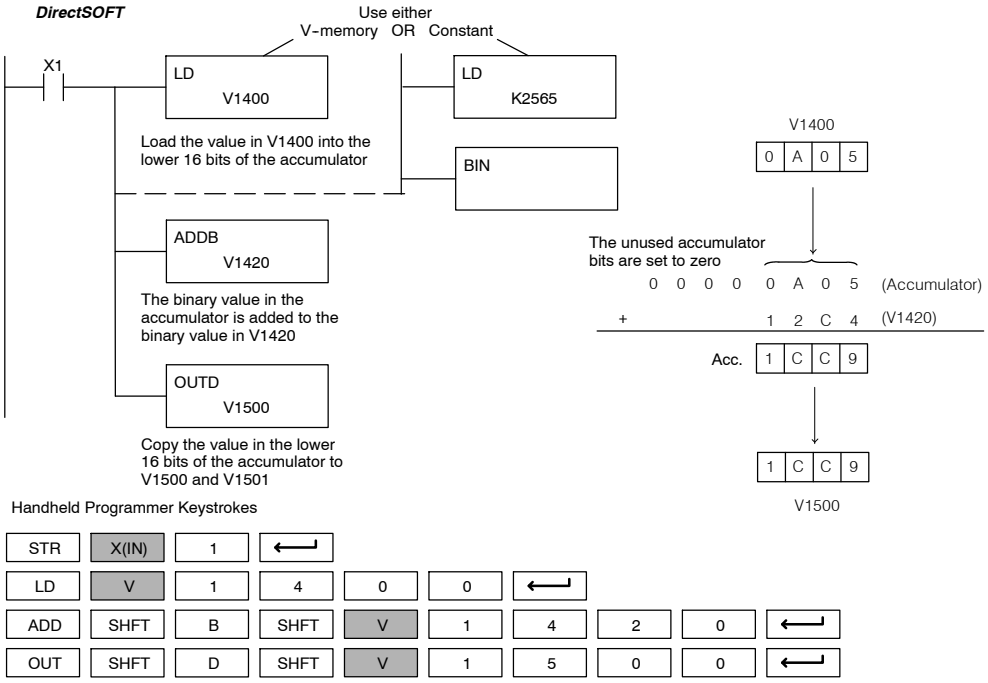


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All V mem (See p. 3-41)	All V mem (See p. 3-42)
Constant	K	0-FFFF	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16 bit addition instruction results in a carry.
SP67	On when the 32 bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP73	On when a signed addition or subtraction results in a incorrect sign bit.

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

In the following example, when X1 is on, the value in V1400 (constant) 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 OUTD instruction.

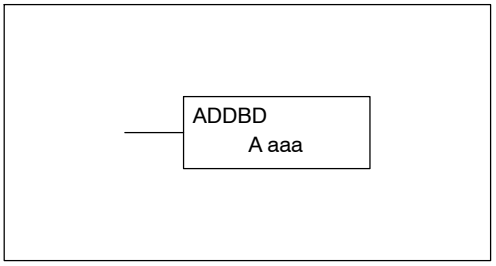


Add Binary Double (ADDBD)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

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.

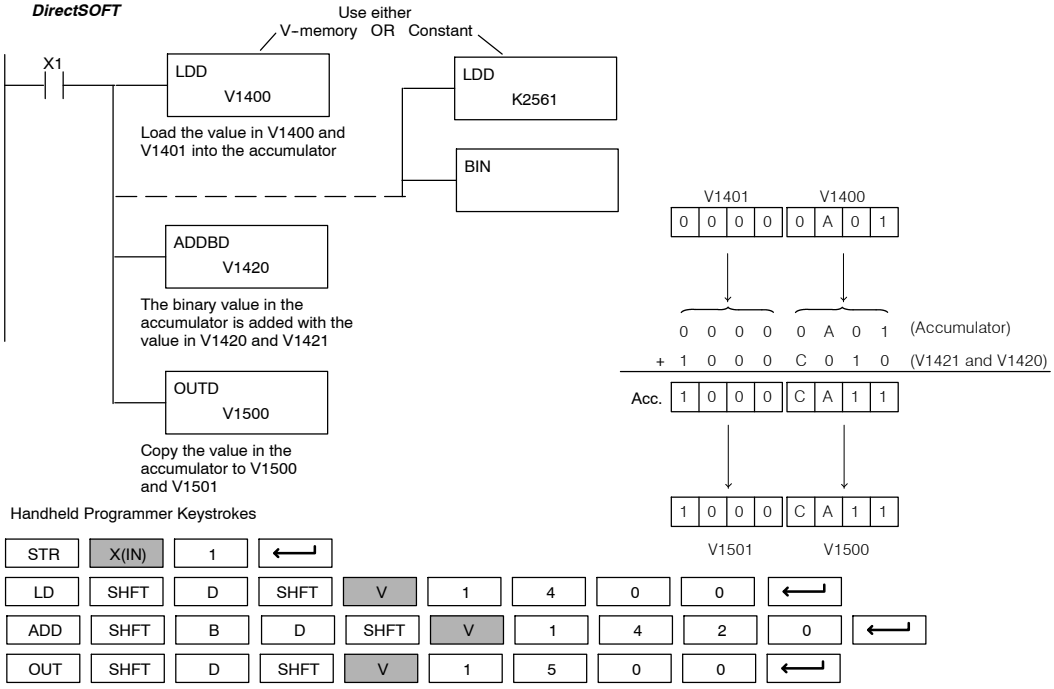


Operand Data Type		DL440 Range	DL450 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-41)	All V mem (See p. 3-42)
Constant	K	0-FFFFFFFF	0-FFFFFFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16 bit addition instruction results in a carry.
SP67	On when the 32 bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP73	On when a signed addition or subtraction results in a incorrect sign bit.

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

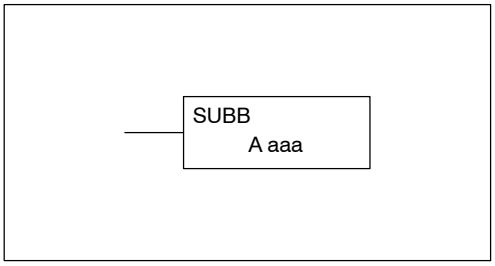
In the following example, when X1 is on, the value in V1400 and V1401 (constant) will be loaded into the accumulator using the LDD instruction. The binary value in the accumulator is added with the binary value in V1420 and V1421 using the ADDBD instruction. The value in the accumulator is copied to V1500 and V1501 using the OUTD instruction.



Subtract Binary (SUBB)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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.

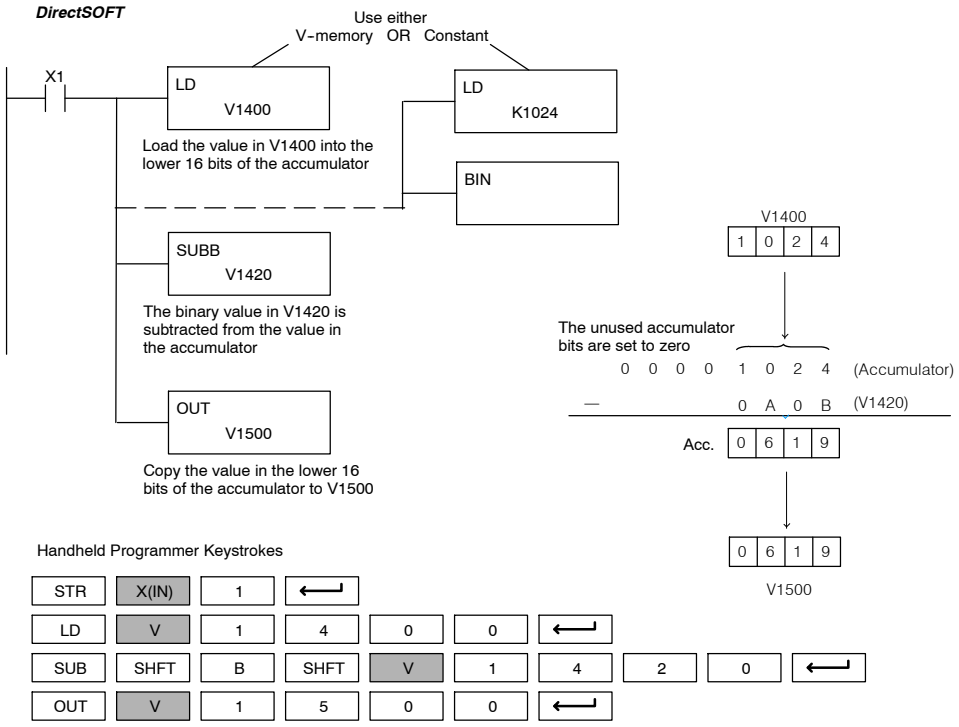


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-FFFF	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16 bit subtraction instruction results in a borrow.
SP65	On when the 32 bit subtraction instruction results in a borrow.
SP70	On anytime the value in the accumulator is negative.

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

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The binary value in V1420 is subtracted from the binary value in the accumulator using the Subtract Binary instruction. The value in the accumulator is copied to V1500 using the Out instruction.

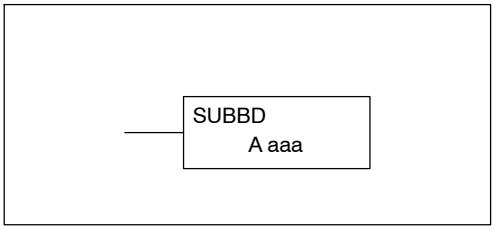


Subtract Binary Double (SUBBD)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

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.

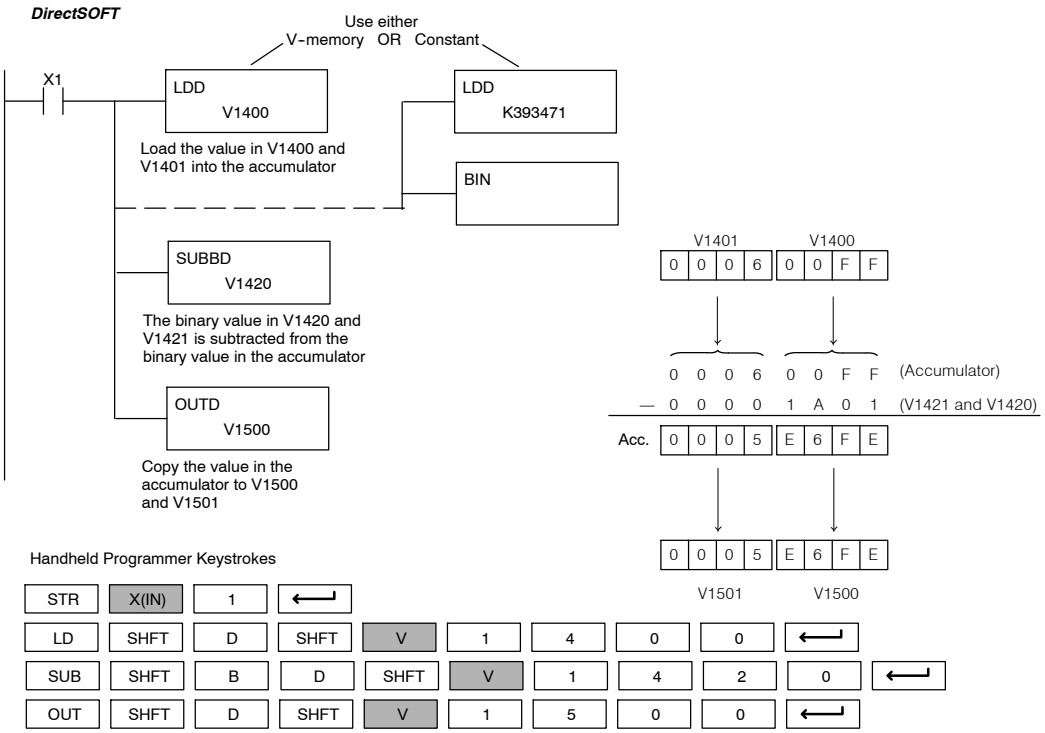


Operand Data Type		DL440 Range	DL450 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-42)	All (See p. 3-42)
Constant	K	0-FFFFFFFF	0-FFFFFFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16 bit subtraction instruction results in a borrow.
SP65	On when the 32 bit subtraction instruction results in a borrow.
SP70	On anytime the value in the accumulator is negative.

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

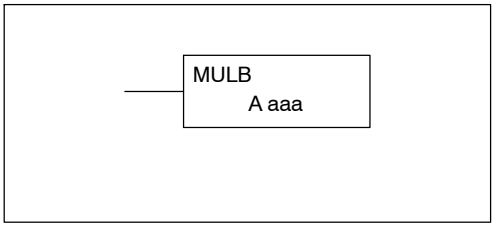
In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The binary value in V1420 and V1421 is subtracted from the binary value in the accumulator using the Subtract Binary Double instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Multiply Binary (MULB)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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.

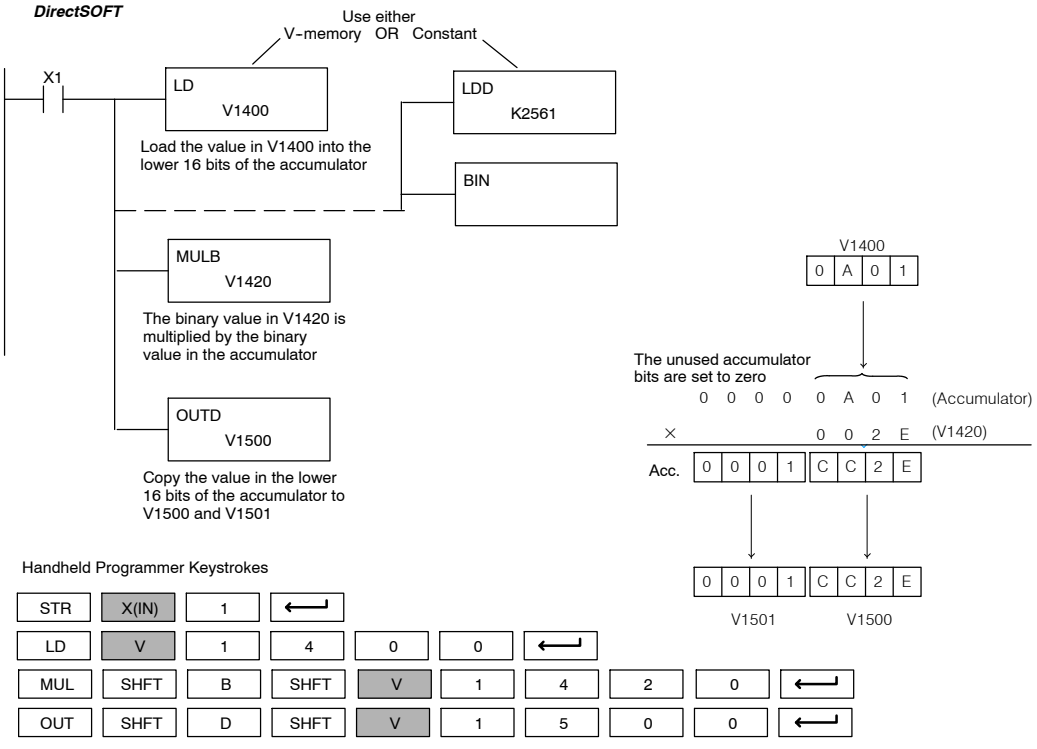


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-FFFF	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.

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

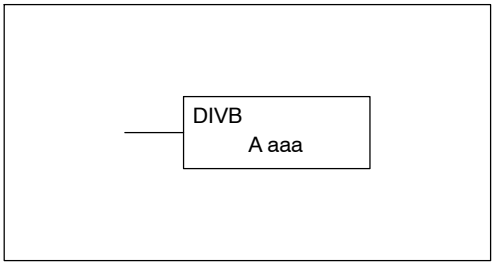
In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The binary value in V1420 is multiplied by the binary value in the accumulator using the Multiply Binary instruction. The value in the accumulator is copied to V1500 using the Out instruction.



Divide Binary (DIVB)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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.

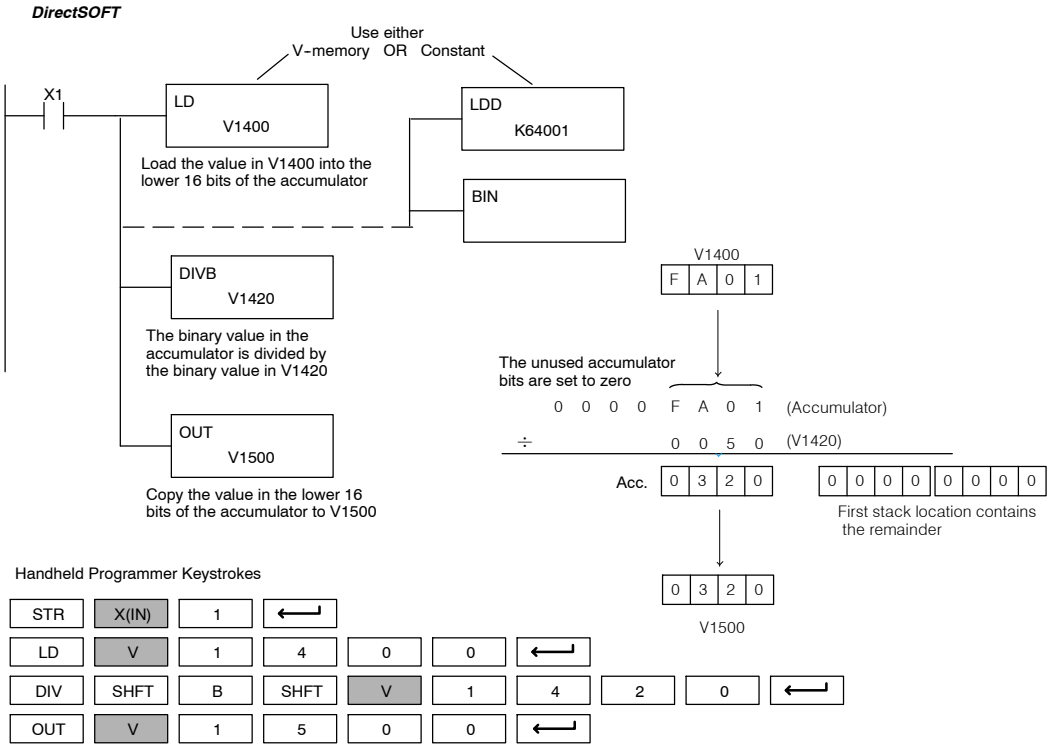


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-FFFF	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.

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

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The binary value in the accumulator is divided by the binary value in V1420 using the Divide Binary instruction. The value in the accumulator is copied to V1500 using the Out instruction.

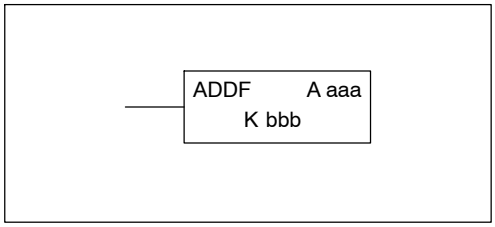


Add Formatted (ADDF)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

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.



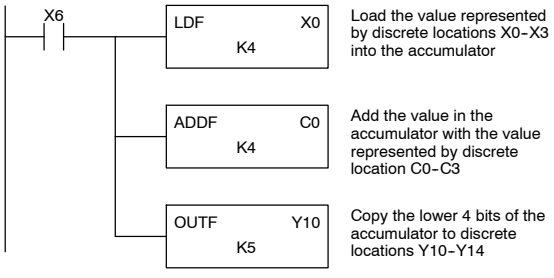
Operand Data Type		DL440 Range		DL450 Range	
	A	aaa	bbb	aaa	bbb
Inputs	X	0-477	--	0-1777	--
Outputs	Y	0-477	--	0-1777	--
Control Relays	C	0-1777	--	0-3777	--
Stage Bits	S	0-1777	--	0-1777	--
Timer Bits	T	0-377	--	0-377	--
Counter Bits	CT	0-177	--	0-377	--
Special Relays	SP	0-137 320-717	--	0-137 320-717	--
Global I/O	GX	0-1777	--	0-2777	--
Constant	K	--	1-32	--	1-32

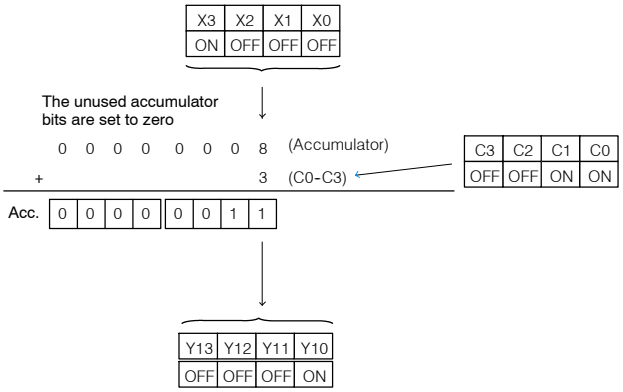
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16 bit addition instruction results in a carry.
SP67	when the 32 bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

In the following example, when X6 is on, the value formed by discrete locations X0-X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete locations C0-C3 is added to the value in the accumulator using the Add Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10-Y13 using the Out Formatted instruction.

DirectSOFT





Handheld Programmer Keystrokes

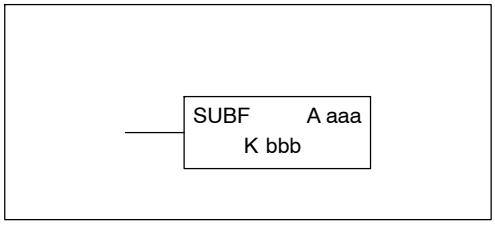
STR	X(IN)	6	↵					
LD	SHFT	F	SHFT	X(IN)	0	K(CON)	4	↵
ADD	SHFT	F	SHFT	C(CR)	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	Y(OUT)	1	0	K(CON)	4 ↵

Subtract Formatted (SUBF)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

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.

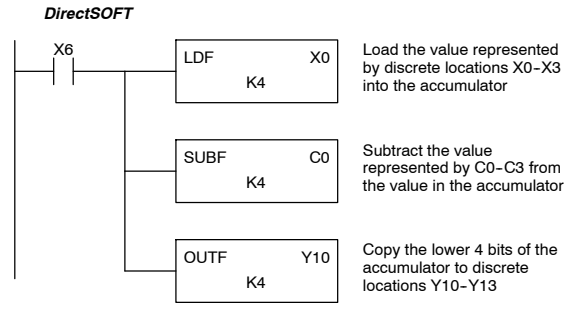


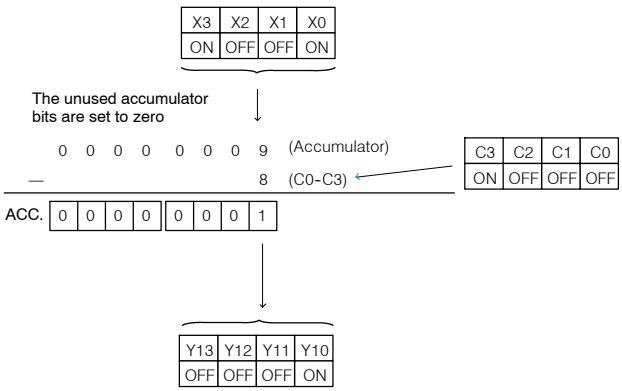
Operand Data Type		DL440 Range		DL450 Range	
	A	aaa	bbb	aaa	bbb
Inputs	X	0-477	--	0-1777	--
Outputs	Y	0-477	--	0-1777	--
Control Relays	C	0-1777	--	0-3777	--
Stage Bits	S	0-1777	--	0-1777	--
Timer Bits	T	0-377	--	0-377	--
Counter Bits	CT	0-177	--	0-377	--
Special Relays	SP	0-137 320-717	--	0-137 320-717	--
Global I/O	GX	0-1777	--	0-2777	--
Constant	K	--	1-32	--	1-32

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16 bit subtraction instruction results in a borrow.
SP65	On when the 32 bit subtraction instruction results in a borrow.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

In the following example, when X6 is on, the value formed by discrete locations X0-X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete location C0-C3 is subtracted from the value in the accumulator using the Subtract Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10-Y13 using the Out Formatted instruction.





Handheld Programmer Keystrokes

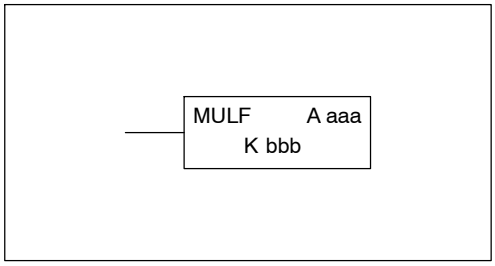
STR	X(IN)	6	↵					
LD	SHFT	F	SHFT	X(IN)	0	K(CON)	4	↵
SUBF	SHFT	F	SHFT	C(CR)	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	Y(OUT)	1	0	K(CON)	4	↵

Multiply Formatted (MULF)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

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.



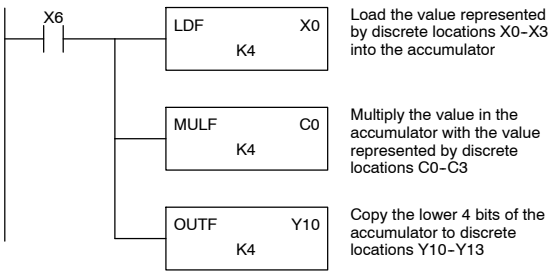
Operand Data Type		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb
Inputs	X	0-477	--	0-1777	--
Outputs	Y	0-477	--	0-1777	--
Control Relays	C	0-1777	--	0-3777	--
Stage Bits	S	0-1777	--	0-1777	--
Timer Bits	T	0-377	--	0-377	--
Counter Bits	CT	0-177	--	0-377	--
Special Relays	SP	0-137 320-717	--	0-137 320-717	--
Global I/O	GX	0-1777	--	0-2777	--
Constant	K	--	1-16	--	1-16

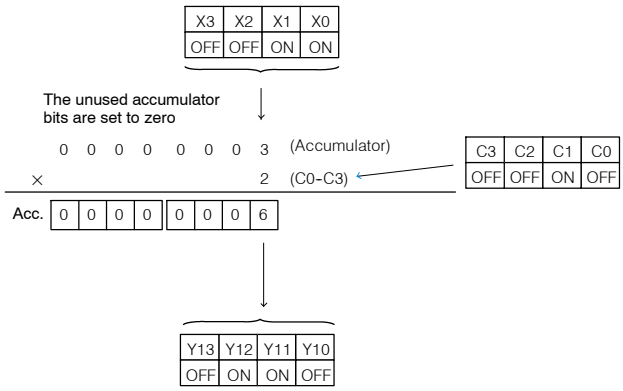
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

In the following example, when X6 is on, the value formed by discrete locations X0-X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete locations C0-C3 is multiplied by the value in the accumulator using the Multiply Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10-Y13 using the Out Formatted instruction.

DirectSOFT





Handheld Programmer Keystrokes

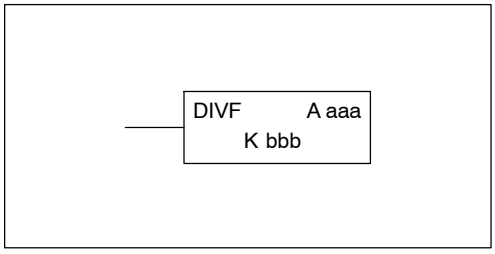
STR	X(IN)	6	↵					
LD	SHFT	F	SHFT	X(IN)	0	K(CON)	4	↵
MUL	SHFT	F	SHFT	C(CR)	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	Y(OUT)	1	0	K(CON)	4	↵

Divide Formatted (DIVF)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

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.



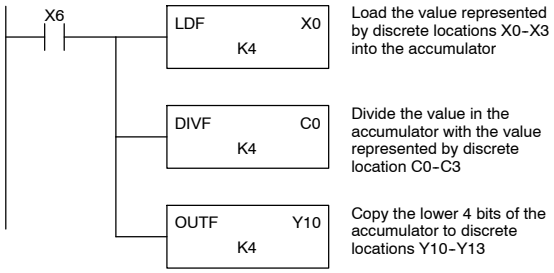
Operand Data Type		DL440 Range		DL450 Range	
	A/B	aaa	bbb	aaa	bbb
Inputs	X	0-477	--	0-477	--
Outputs	Y	0-477	--	0-477	--
Control Relays	C	0-1777	--	0-1777	--
Stage Bits	S	0-1777	--	0-1777	--
Timer Bits	T	0-377	--	0-377	--
Counter Bits	CT	0-177	--	0-177	--
Special Relays	SP	0-137 320-717	--	0-137 320-717	--
Global I/O	GX	0-1777	--	0-1777	--
Constant	K	--	1-16	--	1-16

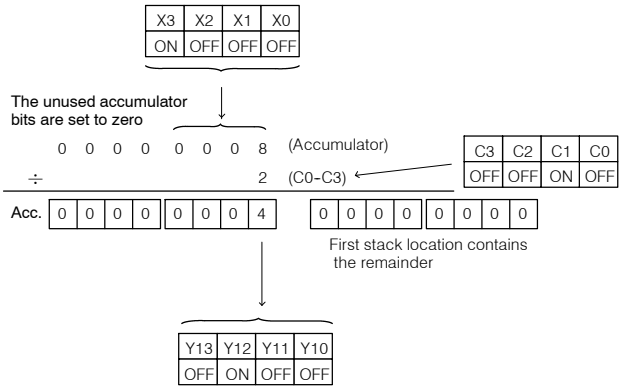
Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

In the following example, when X6 is on, the value formed by discrete locations X0-X3 is loaded into the accumulator using the Load Formatted instruction. The value in the accumulator is divided by the value formed by discrete location C0-C3 using the Divide Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10-Y13 using the Out Formatted instruction.

DirectSOFT





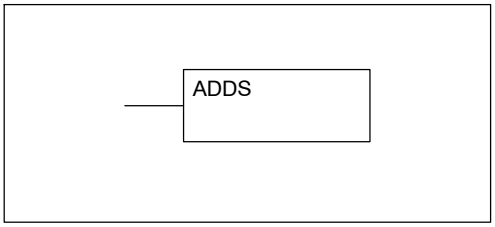
Handheld Programmer Keystrokes

STR	X(IN)	6	↵					
LD	SHFT	F	SHFT	X(IN)	0	K(CON)	4	↵
DIV	SHFT	F	SHFT	C(CR)	0	K(CON)	4	↵
OUT	SHFT	F	SHFT	Y(OUT)	1	0	K(CON)	4	↵

Add Top of Stack (ADDS)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

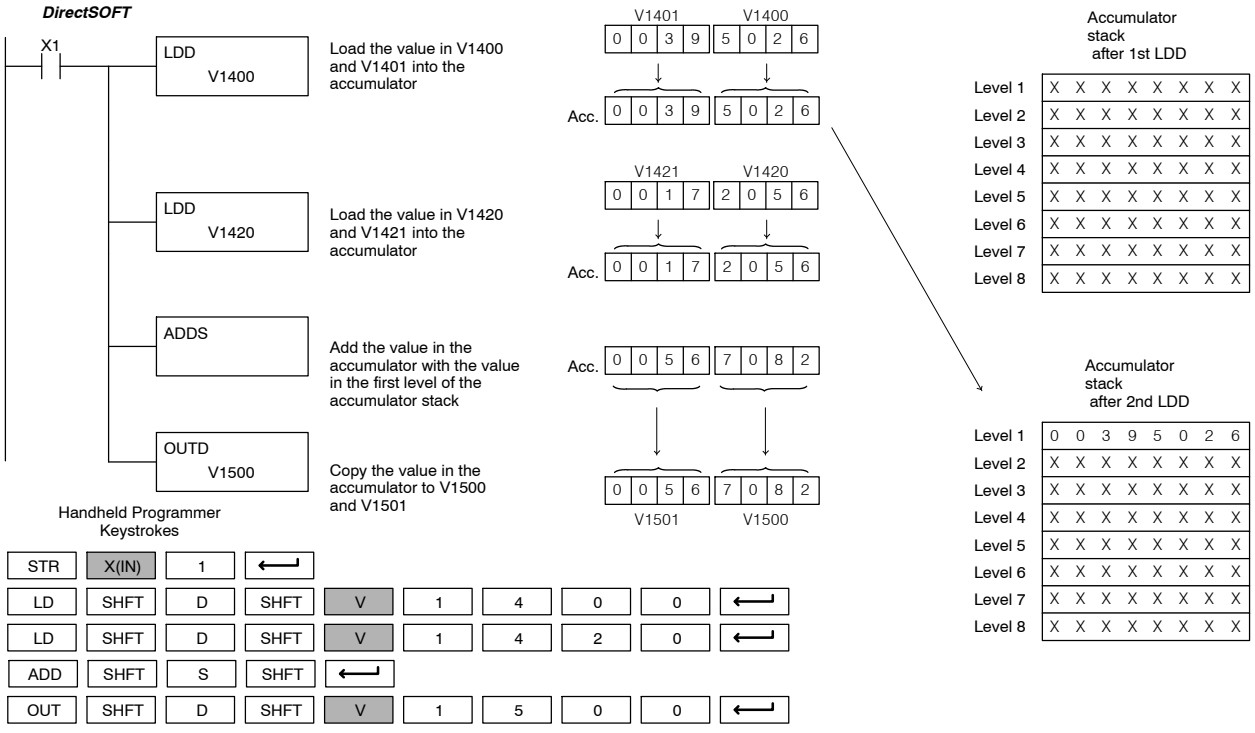
Add Top of Stack is a 32 bit instruction that adds the BCD value in the accumulator with the BCD value in the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack values are moved up one level.



Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16 bit addition instruction results in a 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.

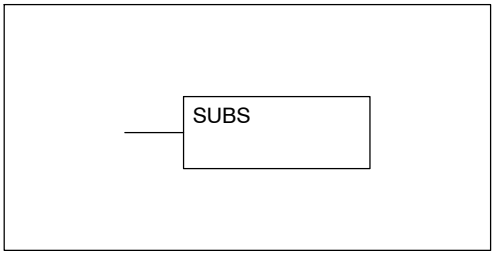
In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator 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.



Subtract Top of Stack (SUBS)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

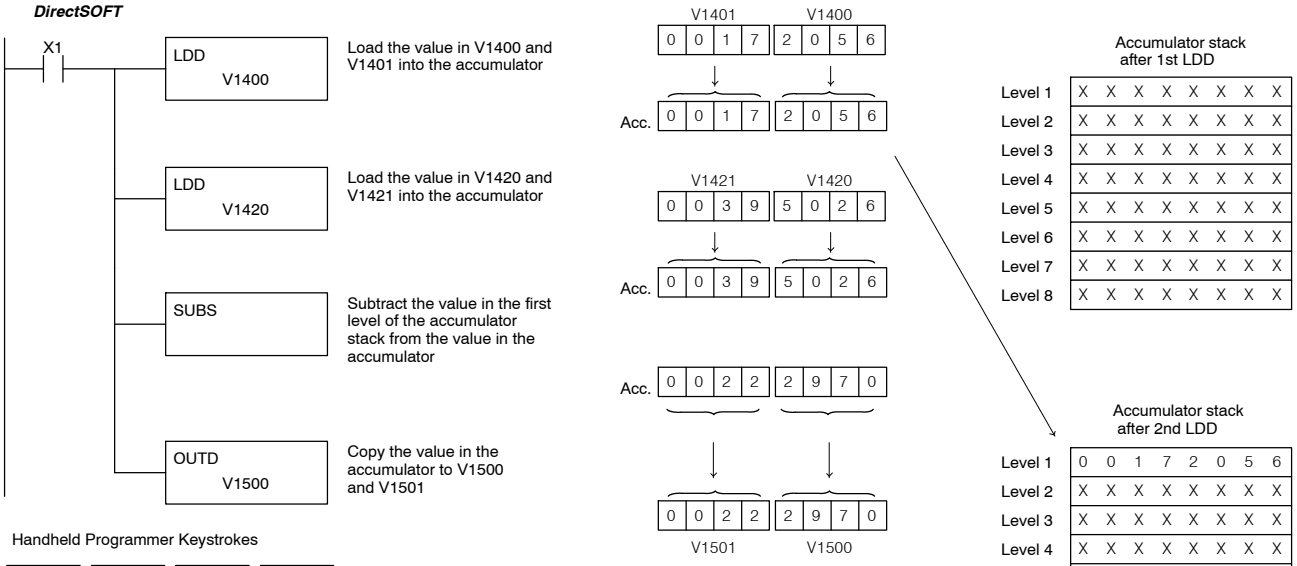
Subtract Top of Stack is a 32 bit instruction that subtracts the BCD value in the first level of the accumulator stack from the BCD value in the accumulator. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack values are moved up one level.



Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16 bit subtraction instruction results in a borrow.
SP65	On when the 32 bit subtraction instruction results in a borrow.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded into the accumulator onto the accumulator stack. The BCD value in the first level of the accumulator stack is subtracted from the BCD value in the accumulator using the Subtract Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

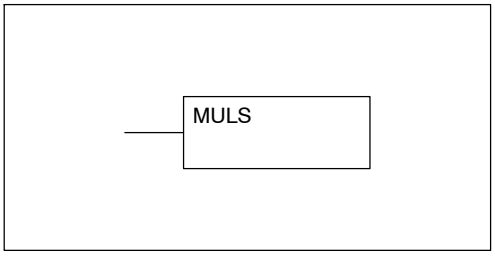
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
LD	SHFT	D	SHFT	V	1	4	2	0	↵
SUB	SHFT	S	SHFT	↵					
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Multiply Top of Stack (MULS)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

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 is removed and all stack values are moved up one level.

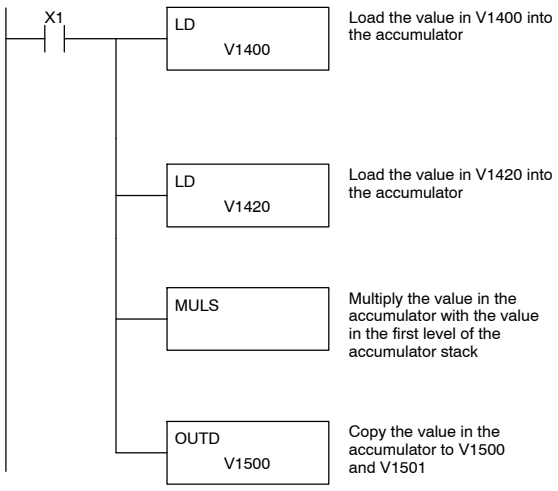


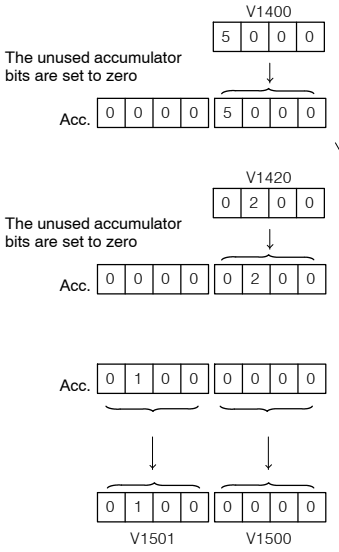
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

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 Double 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.

DirectSOFT





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	5	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

Handheld Programmer Keystrokes

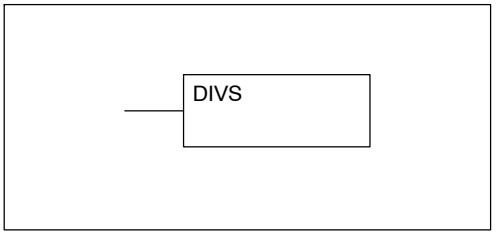
STR	X(IN)	1	↵					
LD	V	1	4	0	0	↵		
LD	V	1	4	2	0	↵		
MUL	SHFT	S	SHFT	↵				
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Divide by Top of Stack (DIVS)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

Divide Top of Stack is a 32 bit instruction that divides the 8-digit BCD value in the accumulator by a 4-digit BCD value in the first level of the accumulator stack. The result resides in the accumulator and the remainder resides in the first level of the accumulator stack.

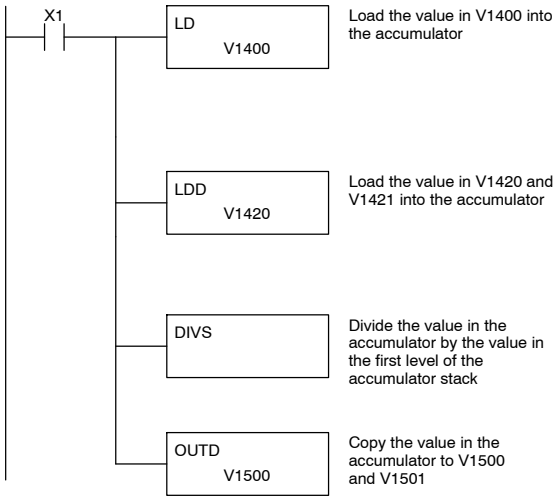


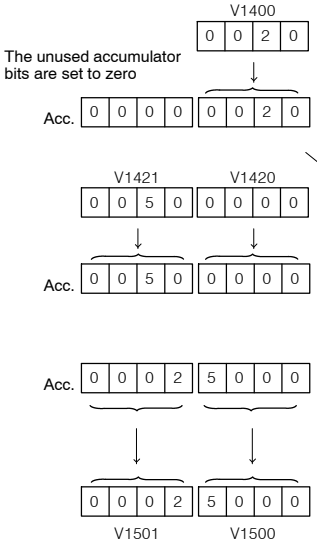
Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

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.

DirectSOFT





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	2	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

Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
LD	SHFT	D	SHFT	V	1	4	2	0	↵
DIV	SHFT	S	SHFT	↵					
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

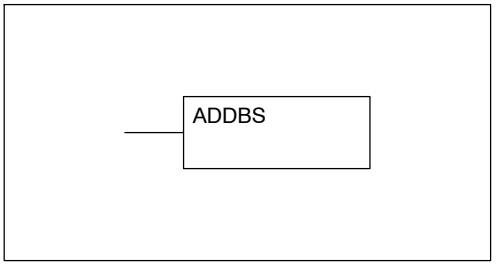
The remainder resides in the first stack location

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

Add Binary Top of Stack (ADDBS)

430	440	450
DS		HPP

Add Binary Top of Stack instruction is a 32 bit instruction that adds the binary value in the accumulator with the binary value in the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack values are moved up one level.

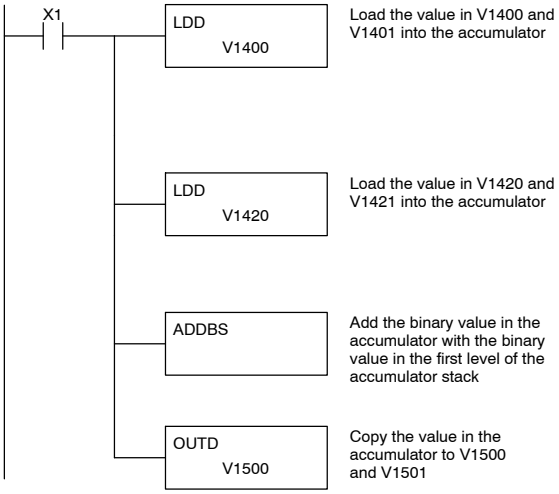


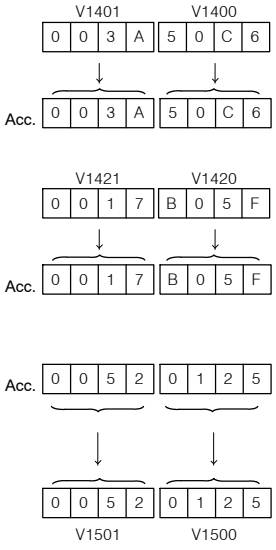
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16 bit addition instruction results in a carry.
SP67	On when the 32 bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP73	on when a signed addition or subtraction results in a incorrect sign bit.

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

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.

DirectSOFT





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	3	A	5	0	C	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

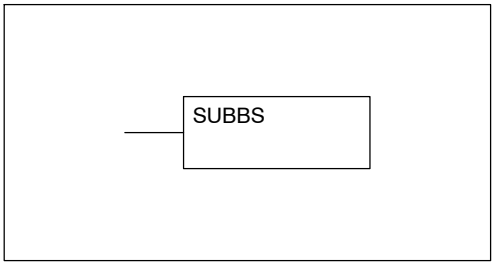
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
LD	SHFT	D	SHFT	V	1	4	2	0	↵
ADD	SHFT	B	S	SHFT	↵				
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Subtract Binary Top of Stack (SUBBS)

430	440	450
DS		HPP

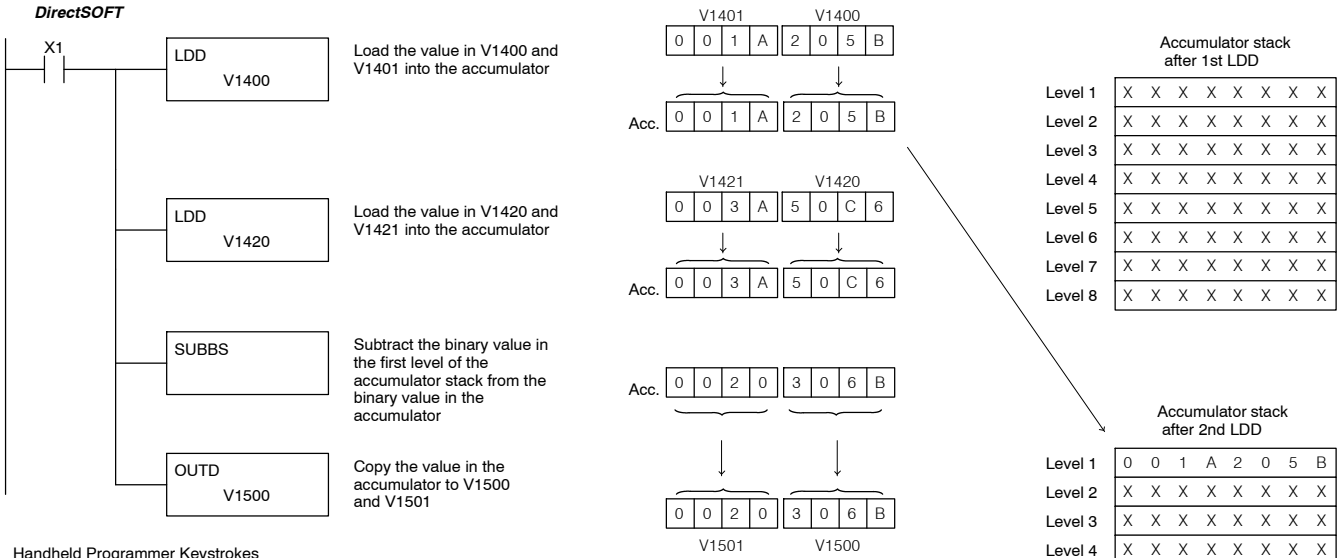
Subtract Binary Top of Stack is a 32 bit instruction that subtracts the binary value in the first level of the accumulator stack from the binary value in the accumulator. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack locations are 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.
SP64	On when the 16 bit subtraction instruction results in a borrow.
SP65	On when the 32 bit subtraction instruction results in a borrow.
SP70	On anytime the value in the accumulator is negative.

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

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The binary value in the first level of the accumulator stack is subtracted from the binary value in the accumulator using the Subtract Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

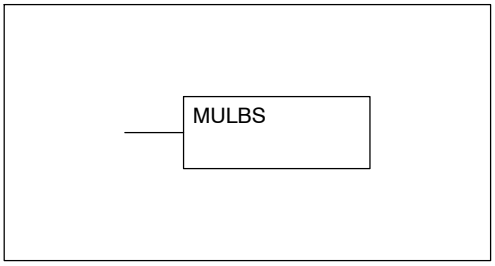
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
LD	SHFT	D	SHFT	V	1	4	2	0	↵
SUB	SHFT	B	S	SHFT	↵				
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Multiply Binary Top of Stack (MULBS)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

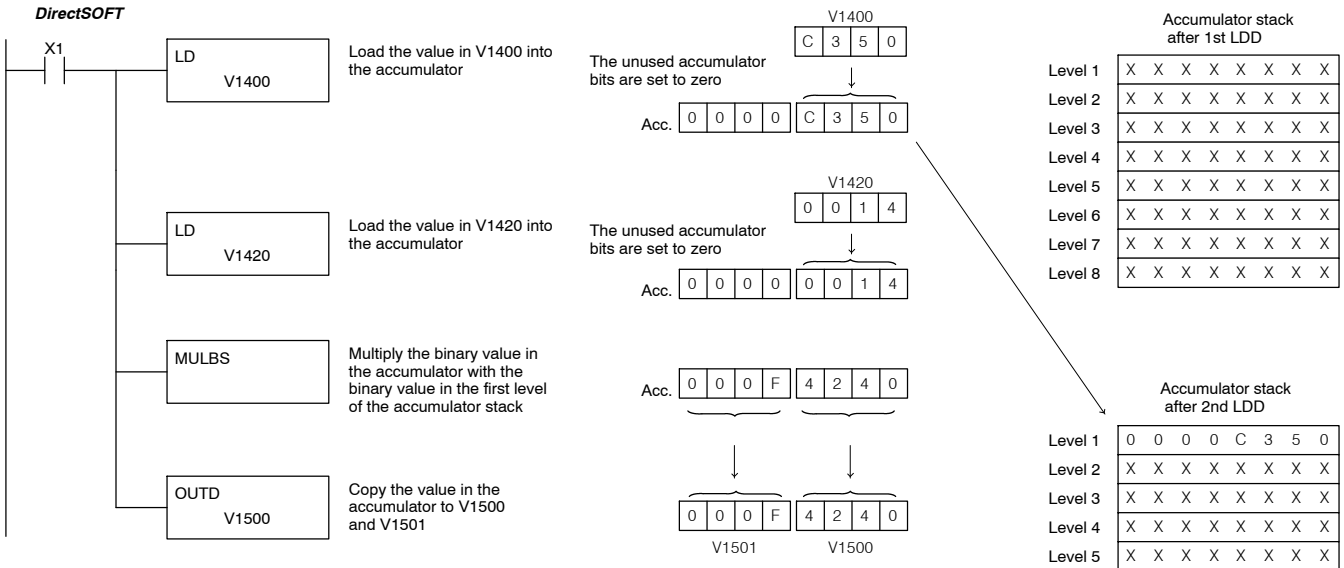
Multiply Binary Top of Stack is a 16 bit instruction that multiplies the 16 bit binary value in the first level of the accumulator stack by the 16 bit binary value in the accumulator. The result resides in the accumulator and can be 32 bits (8 digits max.). The value in the first level of the accumulator 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 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 Load instruction moves the value in V1400 into the accumulator. The value in V1420 is loaded into the accumulator using the Load instruction, pushing the value previously loaded in the accumulator onto the stack. The binary value in the accumulator stack's first level is multiplied by the binary value in the accumulator using the Multiply Binary Stack instruction. The Out Double instruction copies the value in the accumulator to V1500 and V1501.



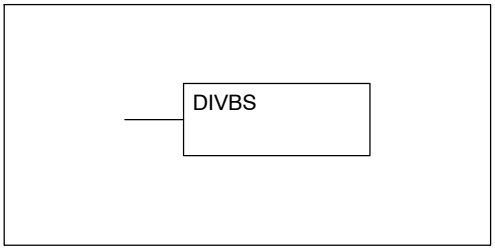
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
LD	V	1	4	2	0	↵			
MUL	SHFT	B	S	SHFT	↵				
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Divide Binary by Top OF Stack (DIVBS)

430	440	450
✓		✓
DS		HPP

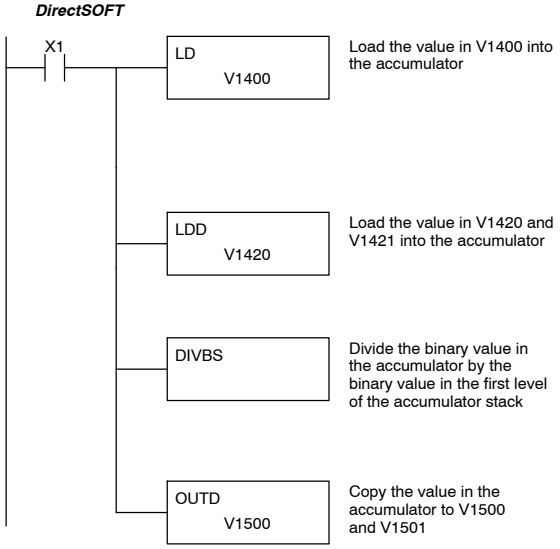
Divide Binary Top of Stack is a 32 bit instruction that divides the 32 bit binary value in the accumulator by the 16 bit binary value in the first level of the accumulator stack. The result resides in the accumulator and the remainder resides in the first level of the accumulator stack.

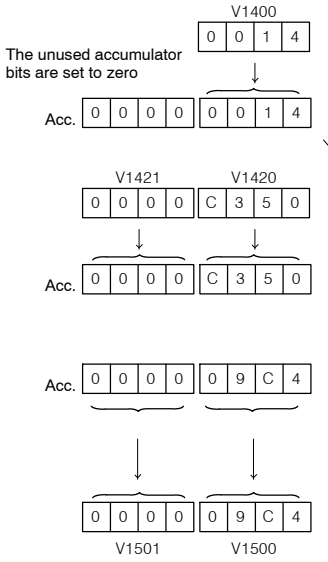


Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.

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

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.





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

Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	V	1	4	0	0	↵			
LD	SHFT	D	SHFT	V	1	4	2	0	↵
DIV	SHFT	B	S	SHFT	↵				
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

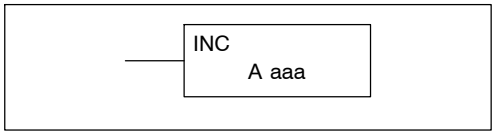
The remainder resides in the first stack location

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

Increment (INC)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

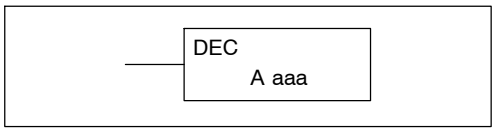
The Increment instruction increments a BCD value in a specified V-memory location by "1" each time the instruction is executed.



Decrement (DEC)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

The Decrement instruction decrements a BCD value in a specified V-memory location by "1" each time the instruction is executed.

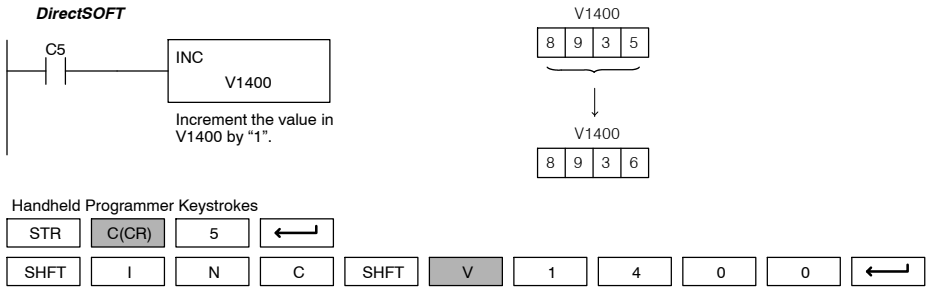


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)

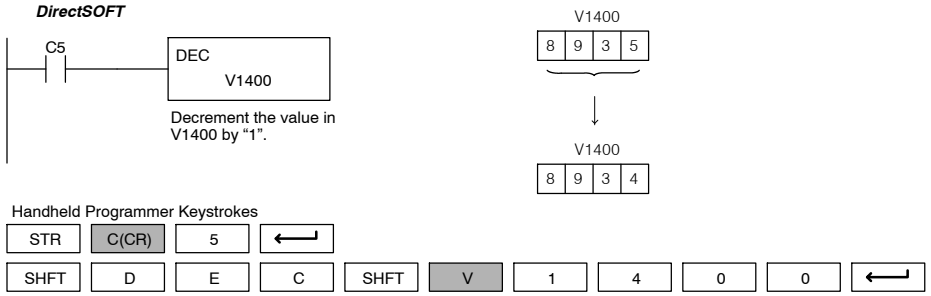
Discrete Bit Flags	Description
SP63	on when the result of the instruction causes the value in the accumulator to be zero.
SP75	on when a BCD instruction is executed and a NON-BCD number was encountered.

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

In the following increment example, when C5 is on the value in V1400 increases by one.



In the following decrement example, when C5 is on the value in V1400 is decreased by one.



Transcendental Functions

The DL450 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 (RAD) instruction, then performing the trig function. All transcendental functions utilize the following flag bits.

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP72	On anytime the value in the accumulator is a valid floating point number.
SP73	on when a signed addition or subtraction results in a incorrect sign bit.
SP75	On when a real number instruction is executed and a non-real number was encountered.

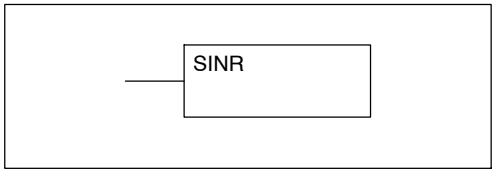
Math Function	Range of Argument
SP53	On when the value of the operand is larger than the accumulator can work with.

Sine Real (SINR)

430	440	450
✗	✗	✓

DS	HPP
✓	✗

The Sine Real instruction takes the sine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

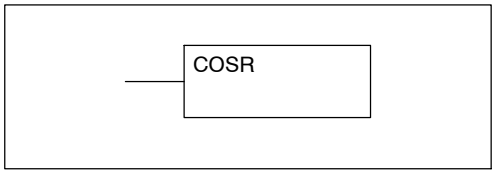


Cosine Real (COSR)

430	440	450
✗	✗	✓

DS	HPP
✓	✗

The Cosine Real instruction takes the cosine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

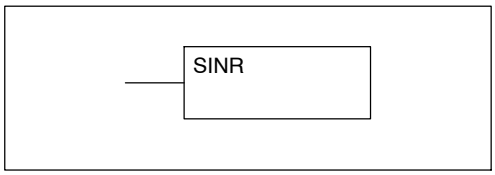


Tangent Real (TANR)

430	440	450
✗	✗	✓

DS	HPP
✓	✗

The Tangent Real instruction takes the tangent of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

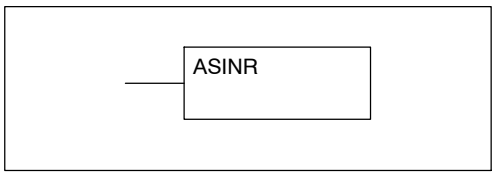


Arc Sine Real (ASINR)

430	440	450
✗	✗	✓

DS	HPP
✓	✗

The Arc Sine Real instruction takes the inverse sine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

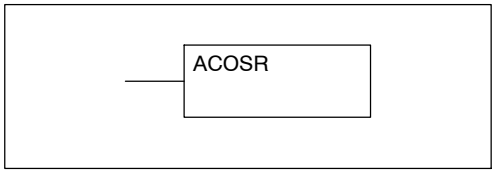


Arc Cosine Real (ACOSR)

430	440	450
✓	✗	✓

DS	HPP
✓	✗

The Arc Cosine Real instruction takes the inverse cosine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

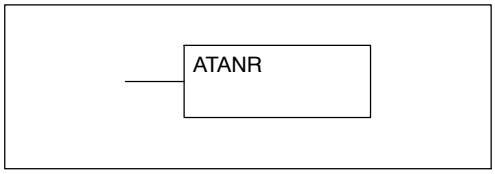


Arc Tangent Real (ATANR)

430	440	450
✓	✗	✓

DS	HPP
✓	✗

The Arc Tangent Real instruction takes the inverse tangent of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

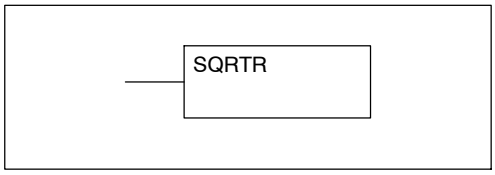


Square Root Real (SQRTR)

430	440	450
✓	✗	✓

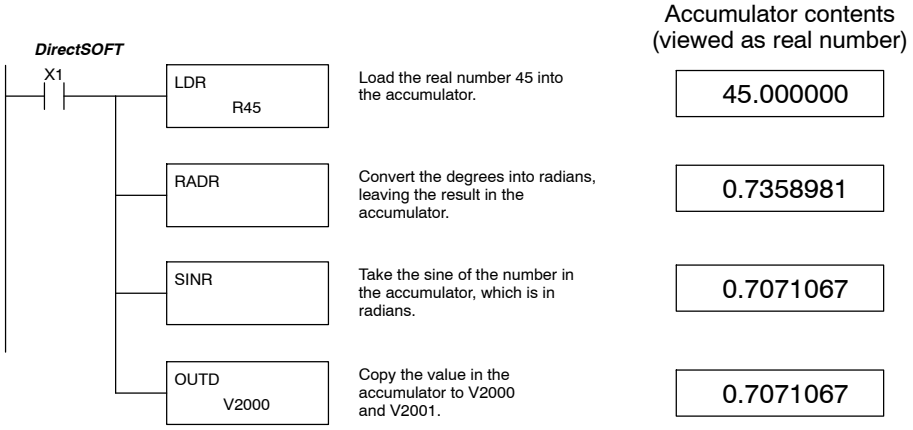
DS	HPP
✓	✗

The Square Root Real instruction takes the square root of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.



NOTE: The square root function can be useful in several situations. However, if you are trying to do the square-root extract function for an orifice flow meter measurement as the PV to a PID loop, note that the PID loop already has the square-root extract function built in.

The following example takes the **sine** of 45 degrees. Since these transcendental functions operate only on real numbers, we do a LDR (load real) 45. The trig functions operate only in radians, so we must convert the degrees to radians by using the RADR command. After using the SINR (Sine Real) instruction, we use an OUTD (Out Double) instruction to move the result from the accumulator to V-memory. The result is 32-bits wide, requiring the Out Double to move it.

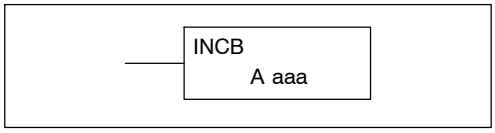


NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use ***Direct***SOFT for entering real numbers, using the LDR (Load Real) instruction.

Increment Binary (INCB)

✓ ✓ ✓
430 440 450
✓ ✓
DS HPP

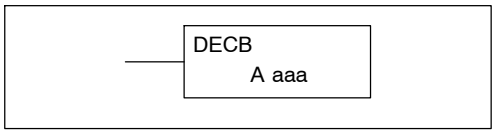
The Increment Binary instruction increments a binary value in a specified V-memory location by "1" each time the instruction is executed.



Decrement Binary (DECB)

✓ ✓ ✓
430 440 450
✓ ✓
DS HPP

The Decrement Binary instruction decrements a binary value in a specified V-memory location by "1" each time the instruction is executed.

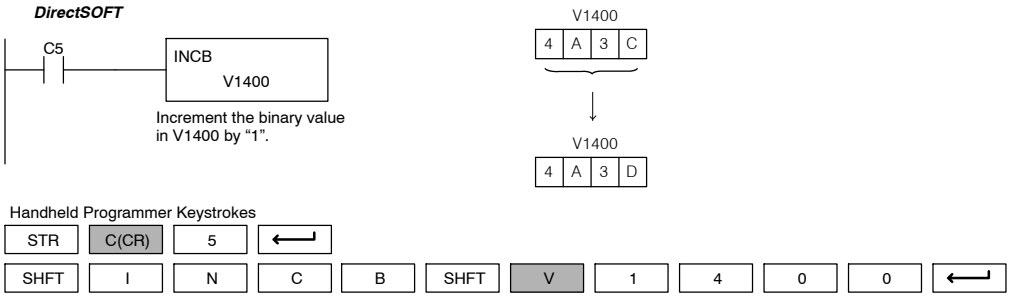


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-40)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)

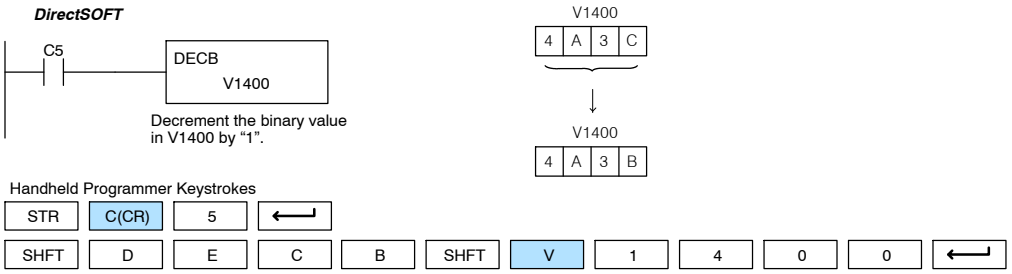
Discrete Bit Flags	Description
SP63	on when the result of the instruction causes the value in the accumulator to be zero.

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

In the following increment binary example, when C5 is on the binary value in V1400 is increased by one.



In the following decrement binary example, when C5 is on the value in V1400 is decreased by one.



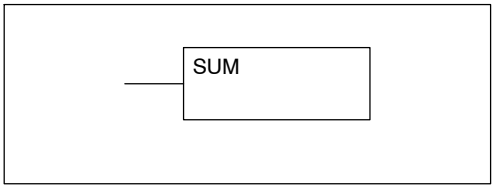
Bit Operation Instructions

Sum (SUM)

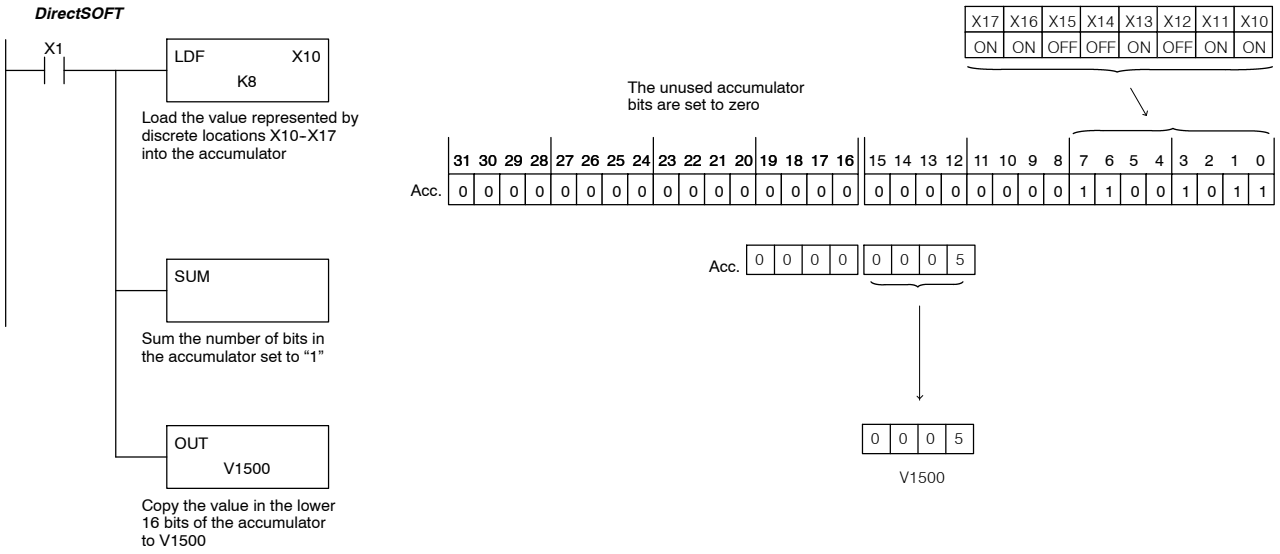
430	440	450
✓	✓	✓

DS	HPP
✓	✓

The Sum instruction counts number of bits that are set to "1" in the accumulator. The HEX result resides in the accumulator.



In the following example, when X1 is on, the value formed by discrete locations X10–X17 is loaded into the accumulator using the Load Formatted instruction. The number of bits in the accumulator set to "1" is counted using the Sum instruction. The value in the accumulator is copied to V1500 using the Out instruction.



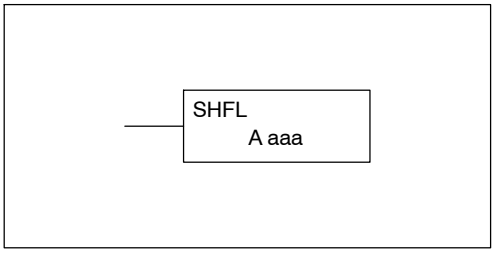
Handheld Programmer Keystrokes

STR	X(IN)	1	↵					
LD	SHFT	F	X(IN)	1	0	K(CON)	8	↵
SHFT	S	U	M	↵				
OUT	V	1	5	0	0	↵		

Shift Left (SHFL)

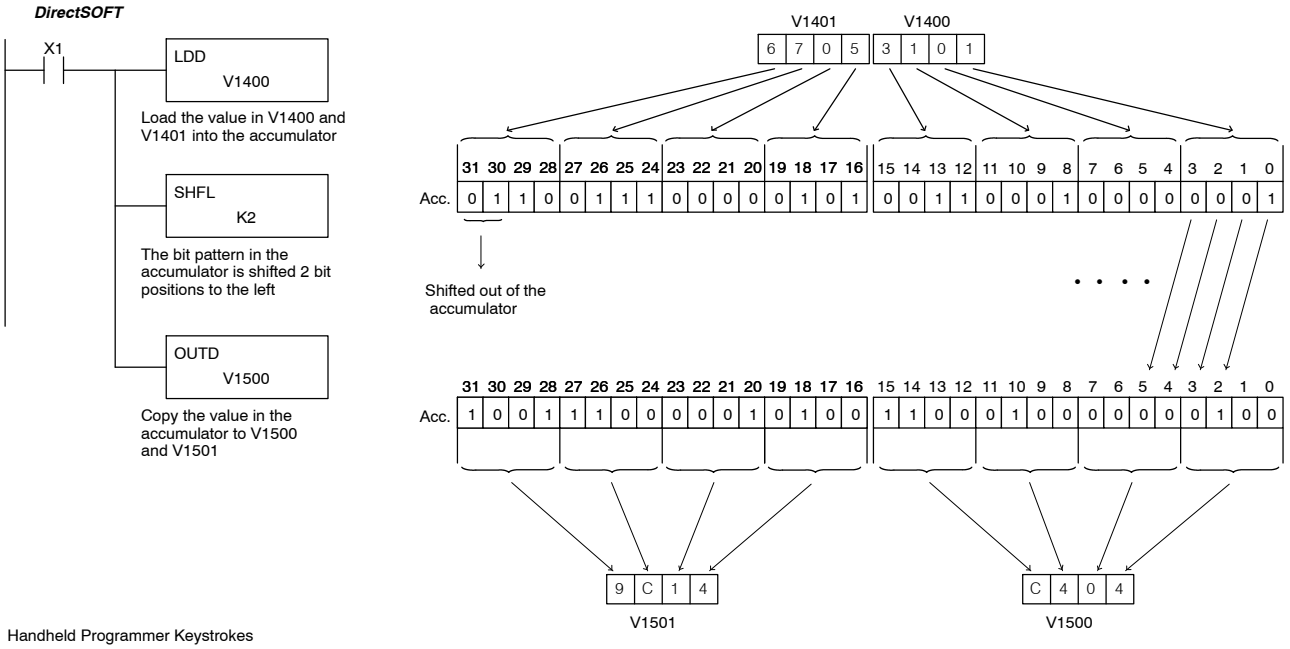
✓ ✓ ✓
430 440 450
✓ ✓
DS HPP

Shift Left is a 32 bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the left. The vacant positions are filled with zeros and the bits shifted out of the accumulator are lost.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Constant	K	1-32	1-32	1-32

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is shifted 2 bits to the left using the Shift Left instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

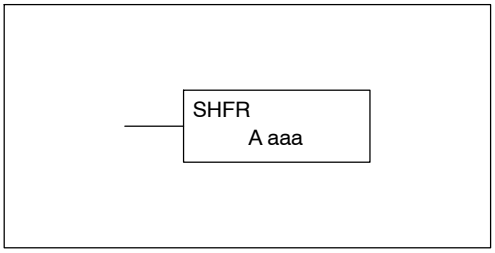
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	S	H	F	L	SHFT	K(CON)	2	↵	
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Shift Right (SHFR)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

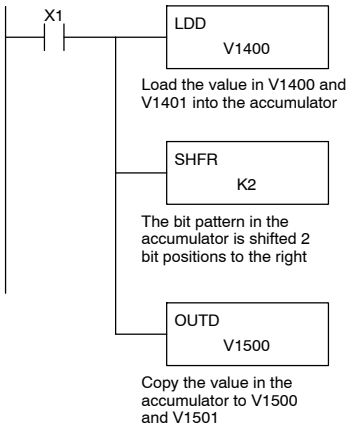
Shift Right is a 32 bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the right. The vacant positions are filled with zeros and the bits shifted out of the accumulator are lost.

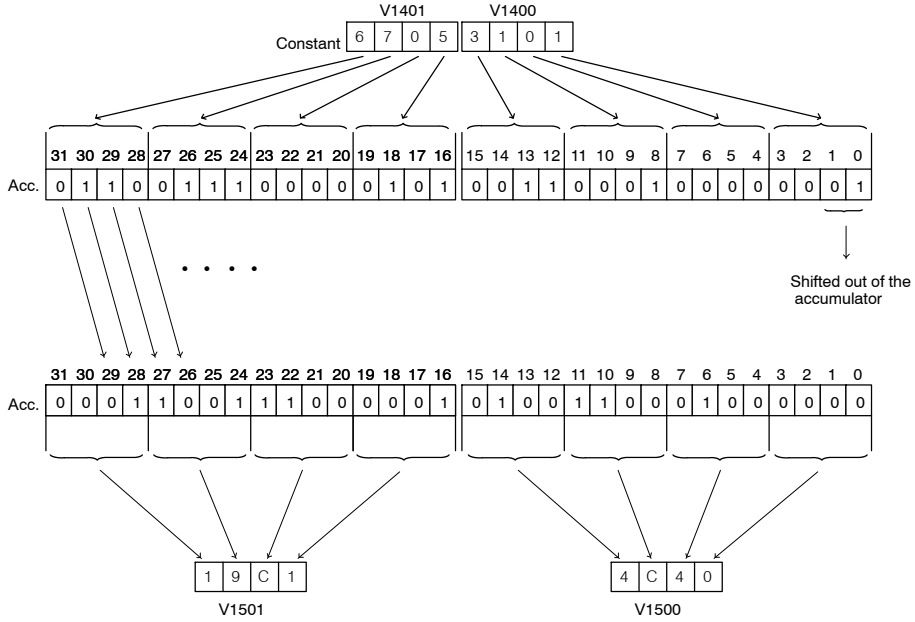


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Constant	K	1-32	1-32	1-32

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is shifted 2 bits to the right using the Shift Right instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

DirectSOFT





Handheld Programmer Keystrokes

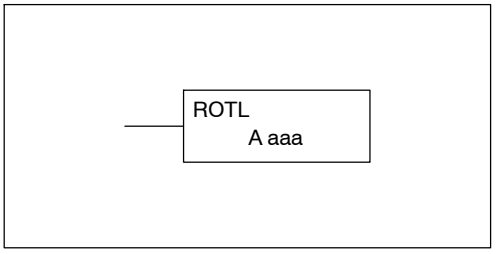
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	S	H	F	R	SHFT	K(CON)	2	↵	
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Rotate Left (ROTL)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

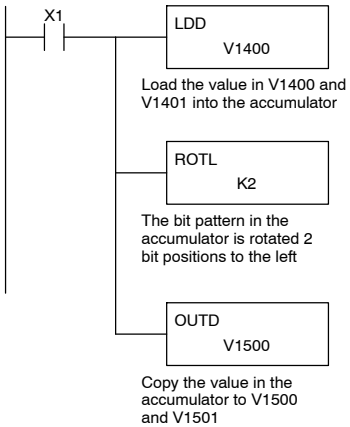
Rotate Left is a 32 bit instruction that rotates the bits in the accumulator a specified number (Aaaa) of places to the left.

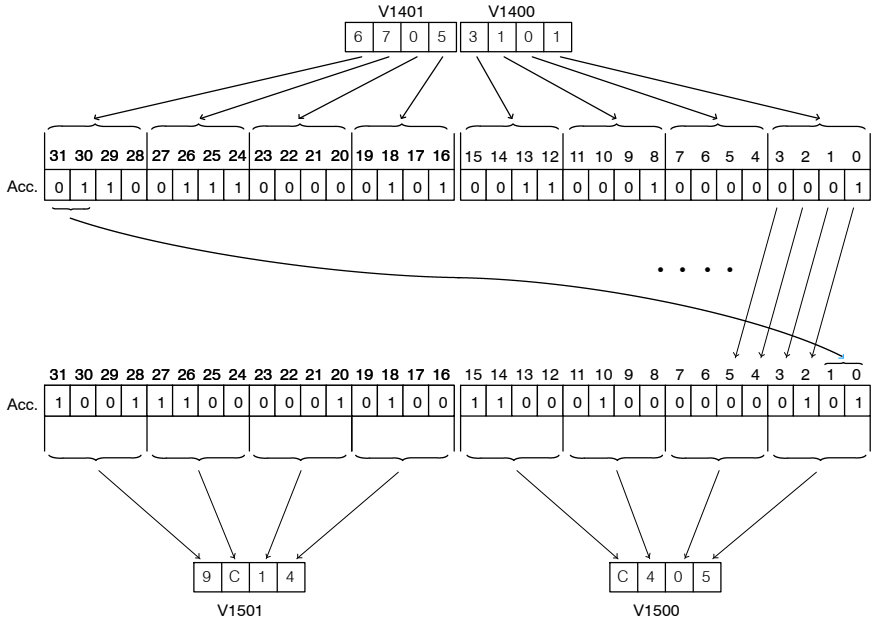


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Constant	K	1-32	1-32	1-32

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is rotated 2 bit positions to the left using the Rotate Left instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

DirectSOFT





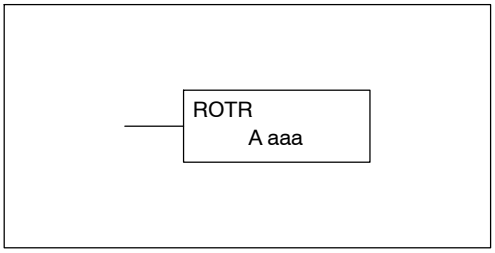
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	R	O	T	L	SHFT	K(CON)	2	↵	
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Rotate Right (ROTR)

✓ ✓ ✓
430 440 450
✓ ✓
DS HPP

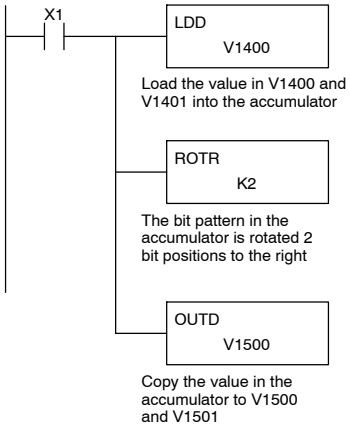
Rotate Right is a 32 bit instruction that rotates the bits in the accumulator a specified number (Aaaa) of places to the right.

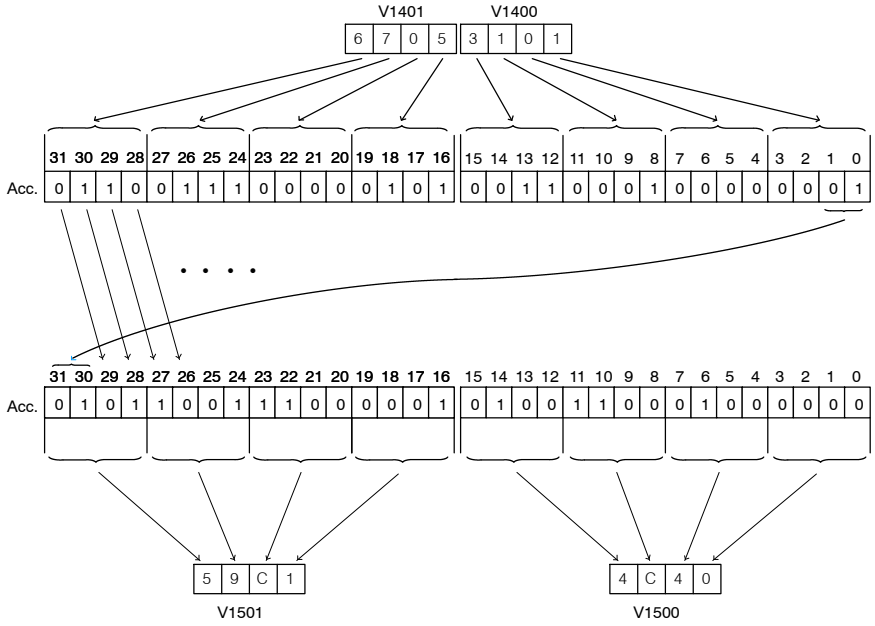


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Constant	K	1-32	1-32	1-32

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is rotated 2 bit positions to the right using the Rotate Right instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

DirectSOFT





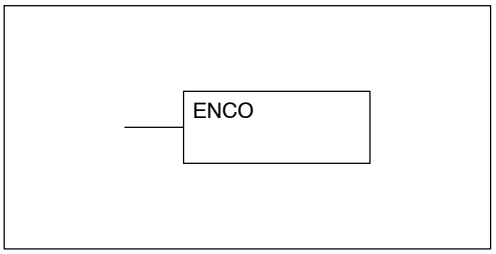
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	R	O	T	R	SHFT	K(CON)	2	↵	
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Encode (ENCO)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

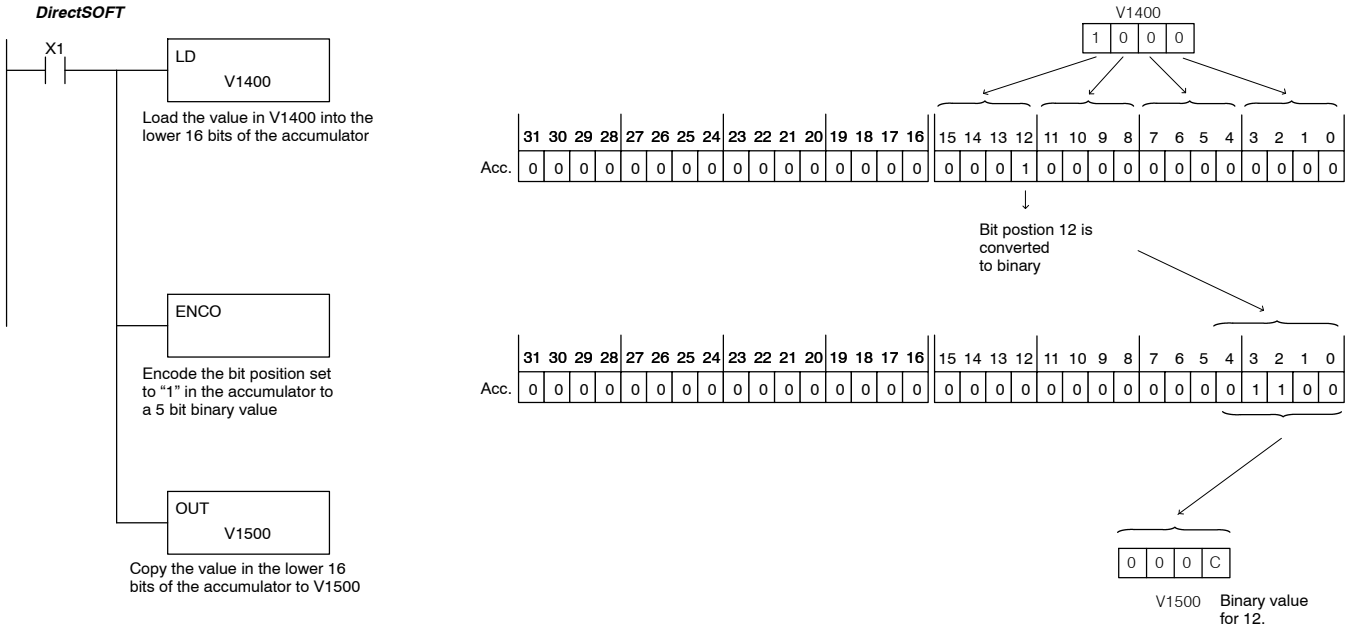
The Encode instruction encodes the bit position in the accumulator having a value of 1, and returns the appropriate binary representation. If the most significant bit is set to 1 (Bit 31), the Encode instruction would place the value HEX 1F (decimal 31) in the accumulator. If the value to be encoded 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. SP53 will also be set on when the accumulator is equal to zero.



Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with. SP53 will also come on when the accumulator is zero.

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, The value in V1400 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 V1500 using the Out instruction.



Handheld Programmer Keystrokes

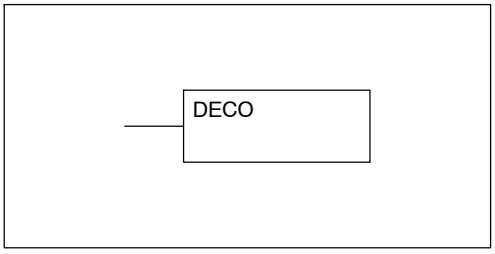
STR	X(IN)	1	↵			
LD	V	1	4	0	0	↵
SHFT	E	N	C	O	↵	
OUT	V	1	5	0	0	↵

Decode (DECO)

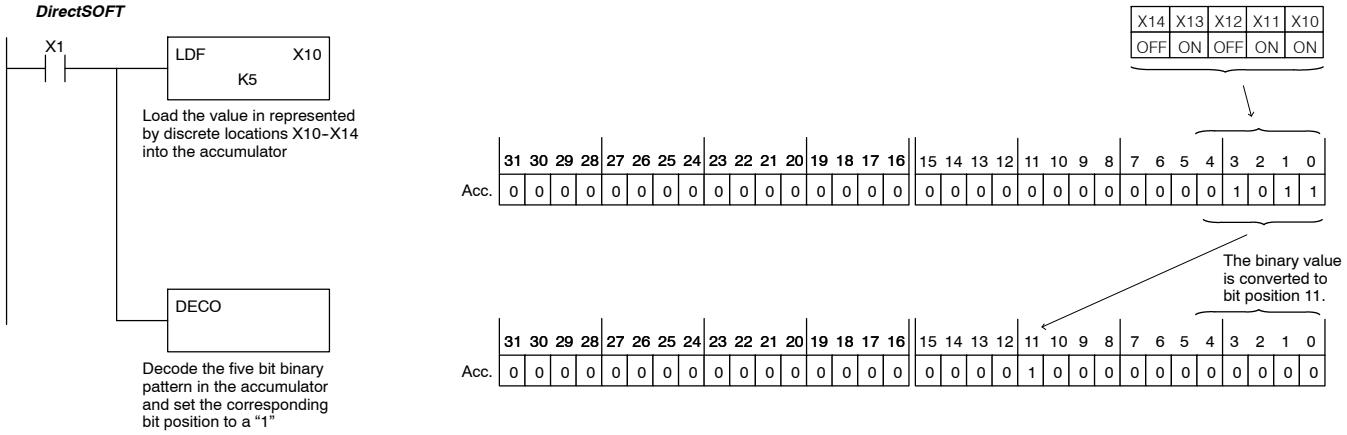
430	440	450
✓	✓	✓

DS	HPP
✓	✓

The Decode instruction decodes a 5 bit binary value of 0–31 (0–1F HEX) in the accumulator by setting the appropriate bit position to a 1. If the accumulator contains the value F (HEX), bit 15 will be set in the accumulator. If the value to be decoded is greater than 31, the number is divided by 32 until the value is less than 32 and then the value is decoded.



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 five bit binary pattern in the accumulator is decoded by setting the corresponding bit position to a "1" using the Decode instruction.



Handheld Programmer Keystrokes

STR	X(IN)	1	↵							
LD	SHFT	F	SHFT	X(IN)	1	0	SHFT	K(CON)	5	↵
SHFT	D	E	C	O	↵					

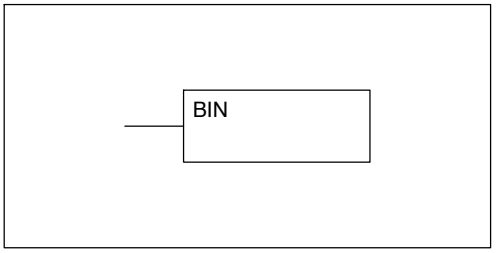
Number Conversion Instructions

Binary (BIN)

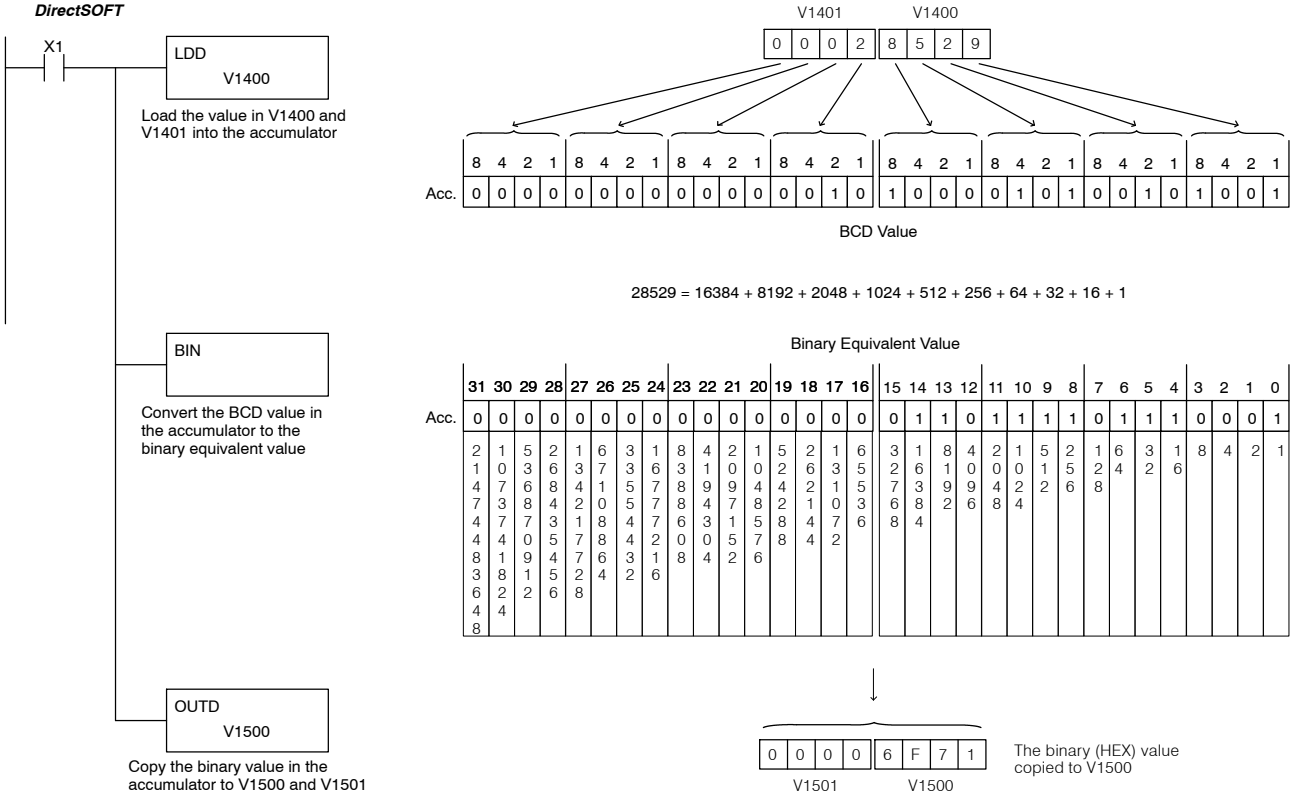
430	440	450
✓	✓	✓

DS	HPP
✓	✓

The Binary instruction converts a BCD value in the accumulator to the equivalent binary value. The result resides in the accumulator.



In the following example, when X1 is on, the value in V1400 and V1401 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 V1500 and V1501 using the Out Double instruction. The handheld programmer would display the binary value in V1500 and V1501 as a HEX value.



Handheld Programmer Keystrokes

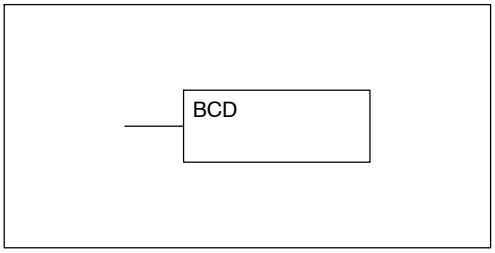
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
BIN	↵								
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Binary Coded Decimal (BCD)

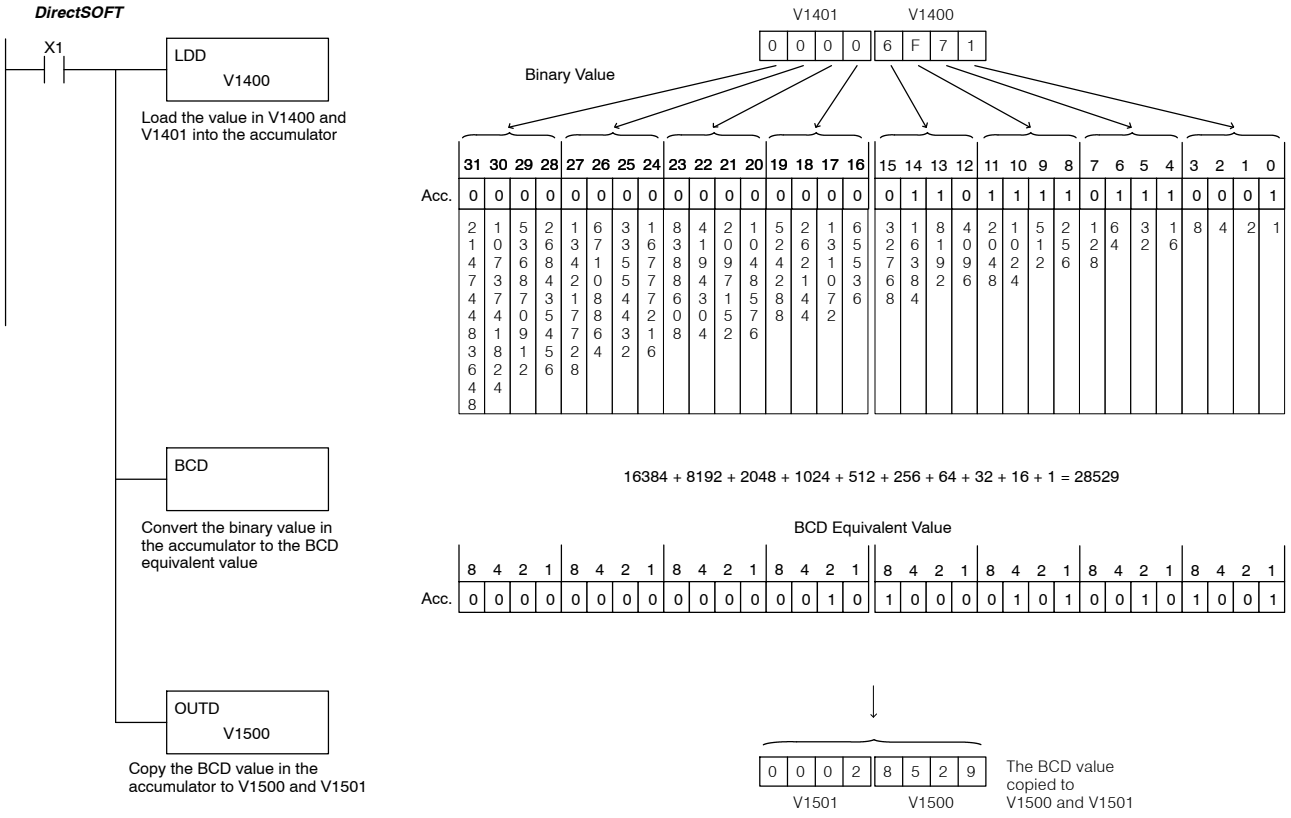
430	440	450
✓	✓	✓

DS	HPP
✓	✓

The Binary Coded Decimal instruction converts a binary value in the accumulator to a BCD value. The result resides in the accumulator.



In the following example, when X1 is on, the binary (HEX) value in V1400 and V1401 is loaded into the accumulator using the Load Double instruction. The binary value in the accumulator is converted to the BCD equivalent value using the BCD instruction. The BCD value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

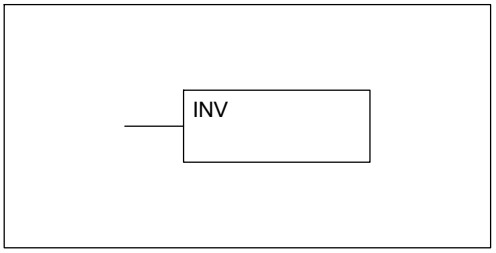
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
BCD	↵								
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Invert (INV)

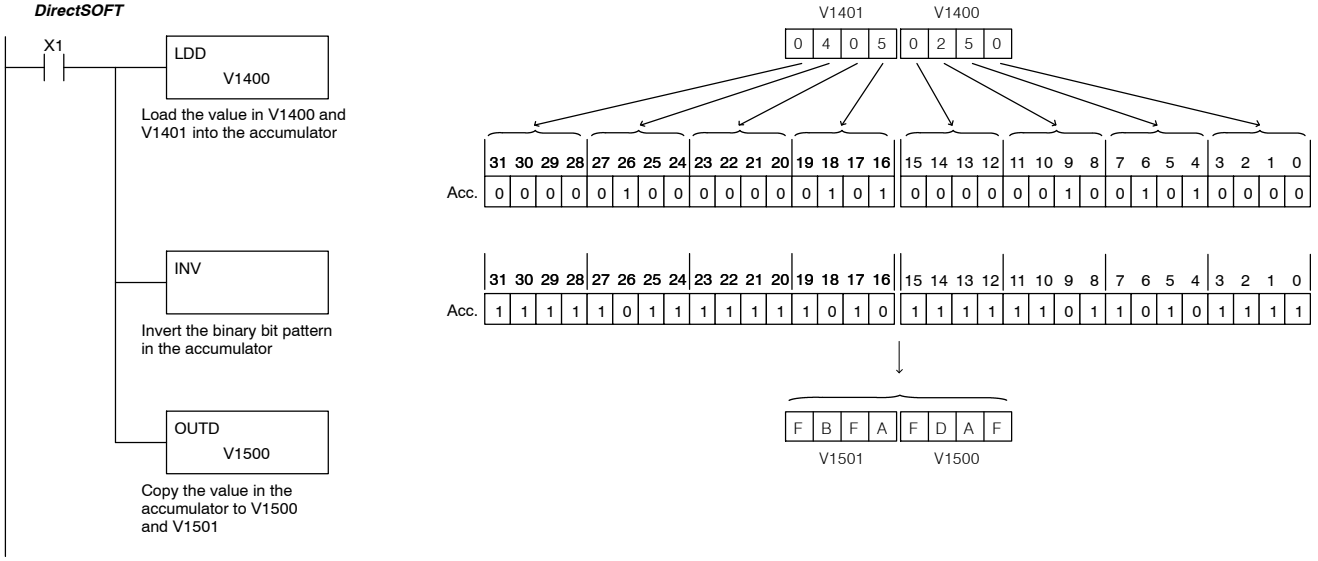
430	440	450
✓	✓	✓

DS	HPP
✓	✓

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 V1400 and V1401 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 V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	I	N	V	↵					
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

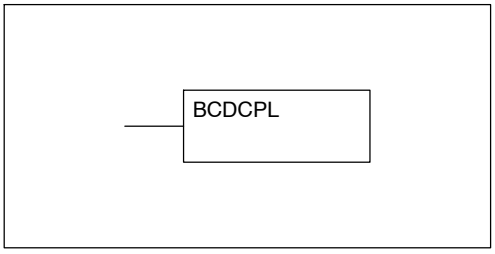
Ten's Complement (BCDCPL)

430	440	450
✓	✓	✓

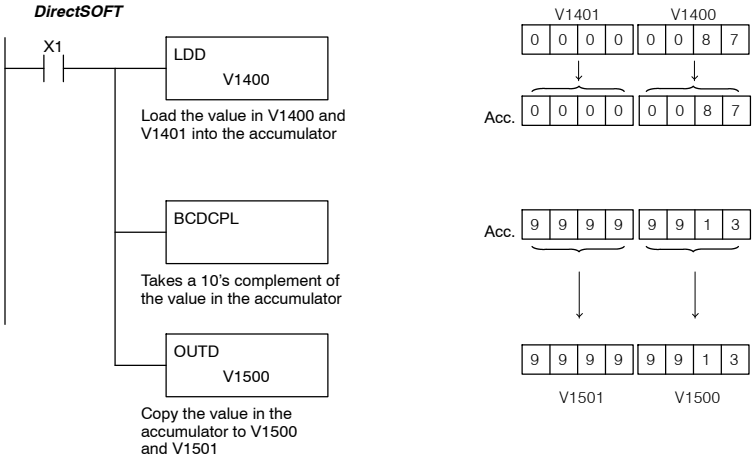
DS	HPP
✓	✓

The Ten's Complement instruction takes the 10's complement (BCD) of the 8 digit accumulator. The result resides in the accumulator. The calculation for this instruction is :

100000000
– accumulator value
10's complement value



In the following example when X1 is on, the value in V1400 and V1401 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 V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
BCD	SHFT	C	P	L	↵				
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

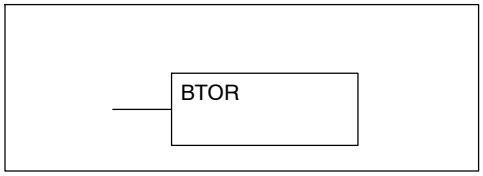
NOTE: If your program has a subtraction calculation which results in a borrowed digit (noted by the status flag), the BCDCPL instruction can be used to find the absolute difference between the two values in the subtraction.

Binary to Real Conversion (BTOR)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

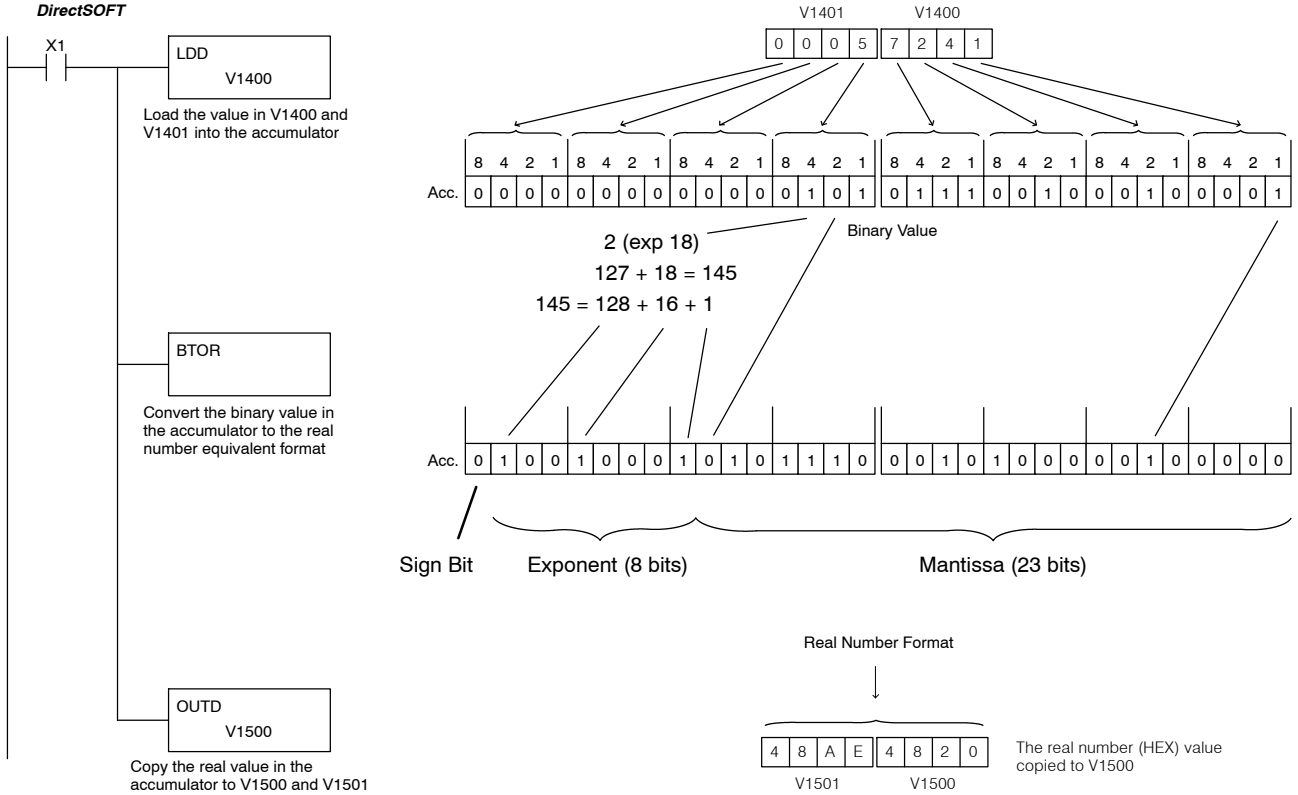
The Binary-to-Real instruction converts a binary value in the accumulator to its equivalent real number (floating point) format. The result resides in the accumulator. Both the binary and the real number may use all 32 bits of the accumulator



NOTE: This instruction will not work for a signed decimal.

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.

In the following example, when X1 is on, the value in V1400 and V1401 is loaded into the accumulator using the Load Double instruction. The BTOR instruction converts the binary value in the accumulator the equivalent real number format. The binary weight of the MSB is converted to the real number exponent by adding it to 127 (decimal). Then the remaining bits are copied to the mantissa as shown. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction. The handheld programmer would display the binary value in V1500 and V1501 as a HEX value.



Handheld Programmer Keystrokes

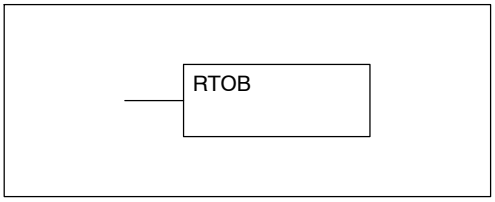
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	B	T	O	R	SHFT	↵			
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Real to Binary Conversion (RTOB)

430	440	450
✗	✗	✓

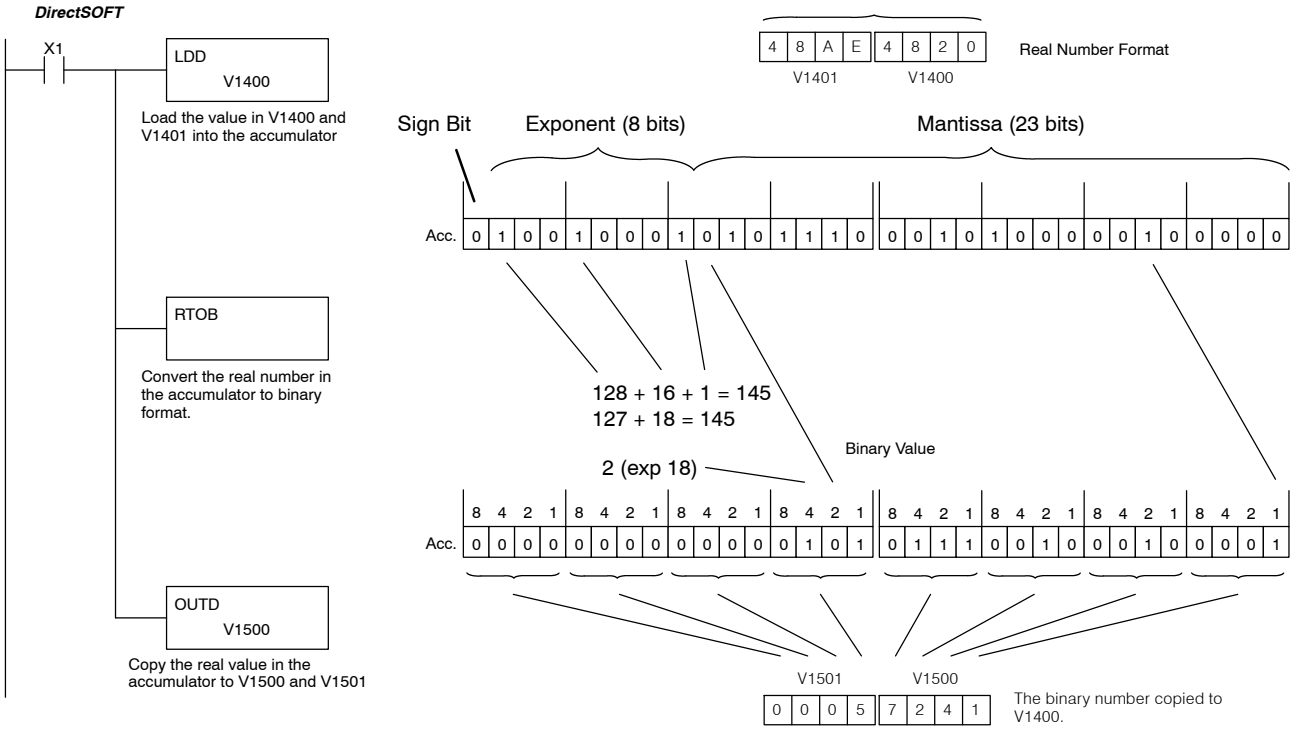
DS	HPP
✓	✓

The Real-to-Binary instruction converts the real number in the accumulator to a binary value. If the real number is negative, it will become a signed decimal. The result resides in the accumulator. Both the binary and the real number may use all 32 bits of the accumulator. Any decimal portion will be truncated.



Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP72	On anytime the value in the accumulator is a valid floating point number.
SP73	on when a signed addition or subtraction results in a incorrect sign bit.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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 to the equivalent binary number format. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction. The handheld programmer would display the binary value in V1500 and V1501 as a HEX value.



Handheld Programmer Keystrokes

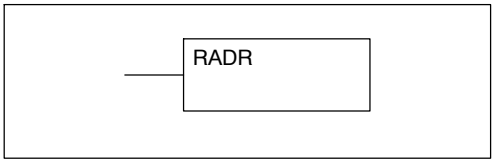
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
SHFT	R	T	O	B	SHFT	↵			
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Radian Real Conversion (RADR)

430	440	450
✗	✗	✓

DS	HPP
✓	✗

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.

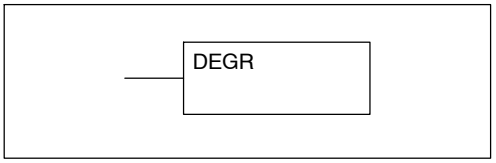


Degree Real Conversion (DEGR)

430	440	450
✗	✗	✓

DS	HPP
✓	✗

The Degree Real instruction converts the degree real radian value stored inthe accumulator to the equivalent real number in degrees. The result resides in the accumulator.

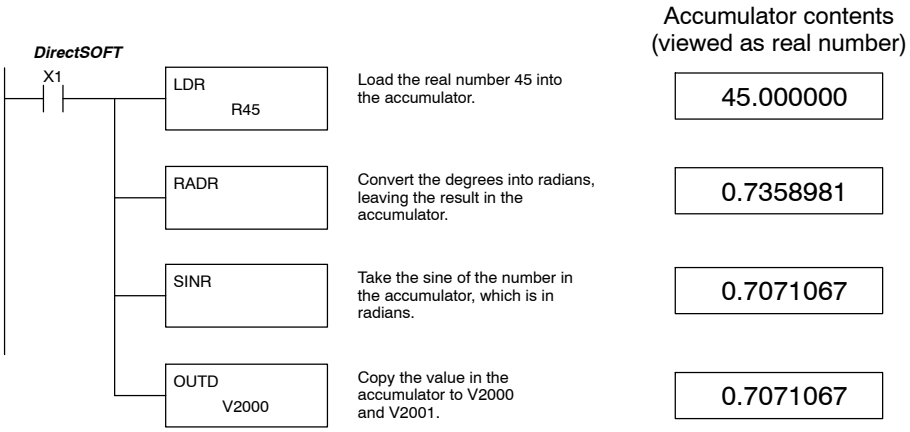


The two instructions described above convert real numbers into the accumulator from degree format to radian format, and visa-versa. In degree format, a circle contains 360 degrees. In radian format, a circle contains 2 Π radians. These convert between both positive and negative real numbers, and for angles greater than a full circle. These functions are very useful when combined with the transcendantal 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.
SP71	On anytime the V-memory specified by a pointer (P) is not valid.
SP72	On anytime the value in the accumulator is a valid floating point number.
SP74	On anytime a floating point math operation results in an underflow error.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use ***Direct***SOFT for entering real numbers, using the LDR (Load Real) instruction.

The following example takes the sine of 45 degrees. Since transcendental functions operate only on real numbers, we do a LDR (load real) 45. The trig functions operate only in radians, so we must convert the degrees to radians by using the RADR command. After using the SINR (Sine Real) instruction, we use an OUTD (Out Double) instruction to move the result from the accumulator to V-memory. The result is 32-bits wide, requiring the Out Double to move it.

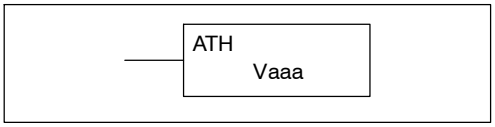


ASCII to HEX (ATH)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

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.

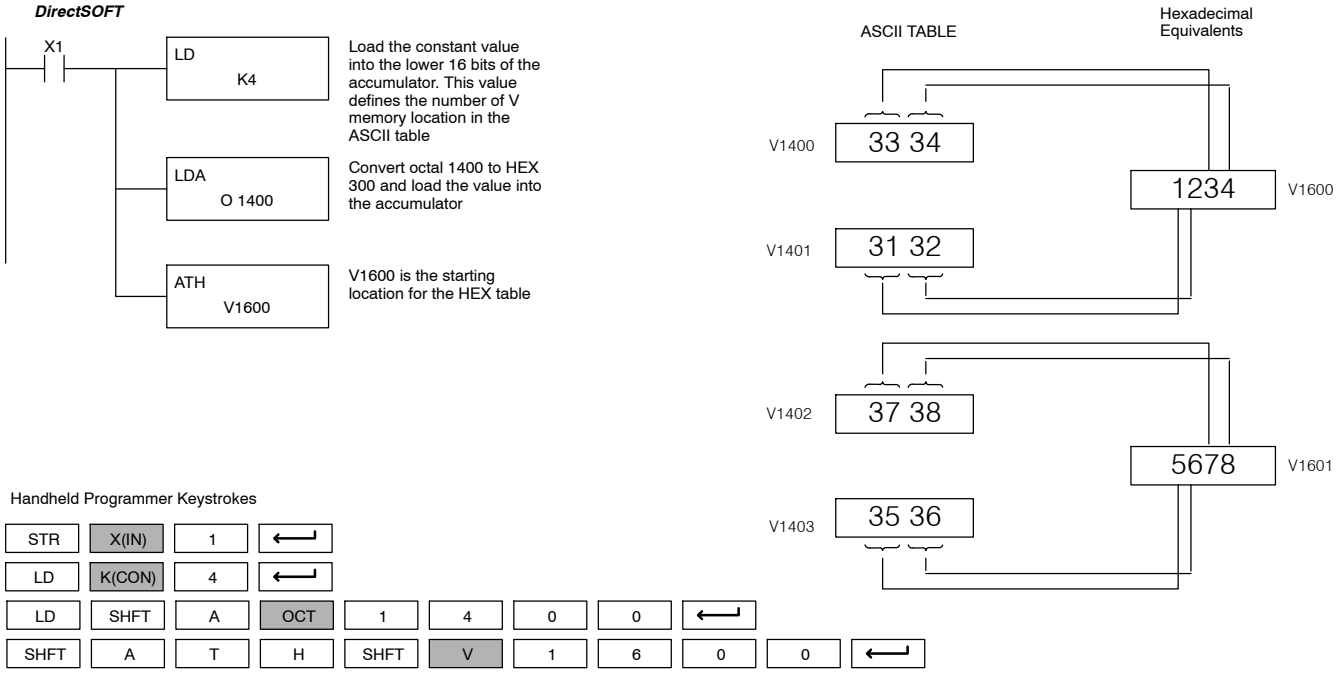
Step 3: — Specify the starting V-memory location (Vaaa) for the HEX table in the ATH instruction.

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		DL440 Range	DL450 Range
		aaa	aaa
V-memory	V	All (See p. 3-41)	All (See p. 3-42)

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

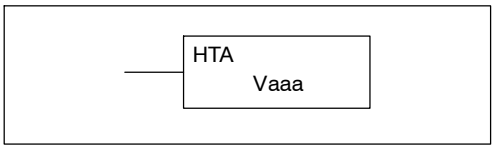


HEX to ASCII (HTA)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

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.

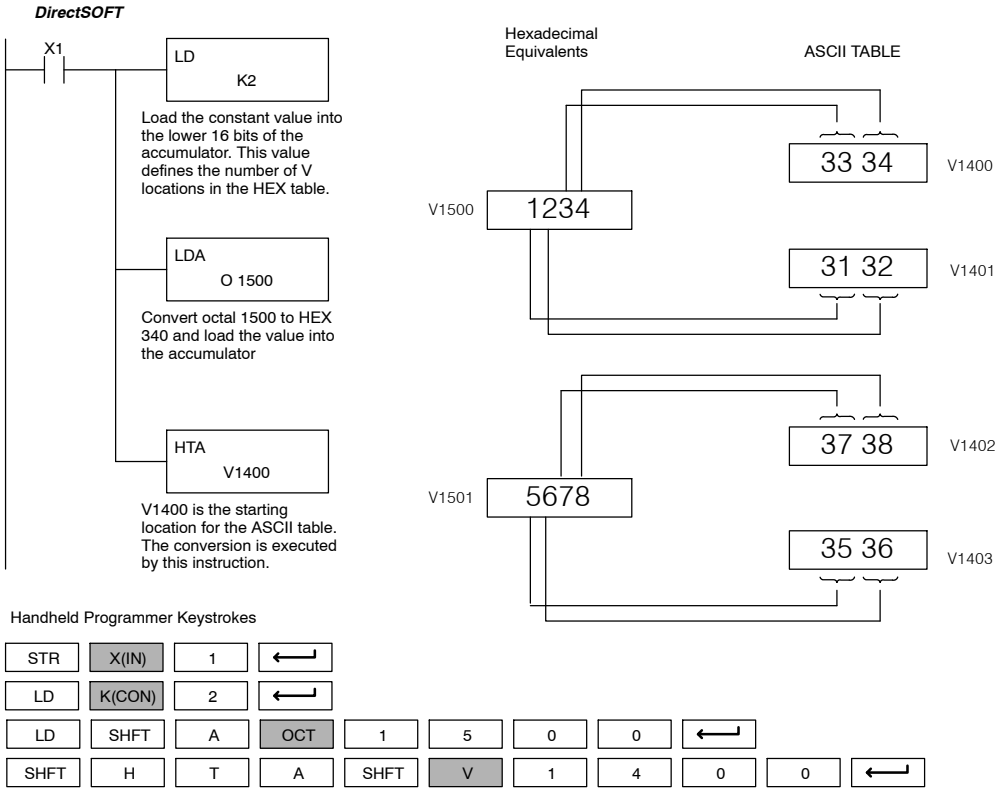
Step 2: — Load the starting V-memory location for the HEX table into the accumulator. This parameter must be a HEX value.

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		DL440 Range	DL440 Range
		aaa	aaa
V-memory	V	All (See p. 3-41)	All (See p. 3-42)

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.



The table below lists valid ASCII values for HTA conversion.

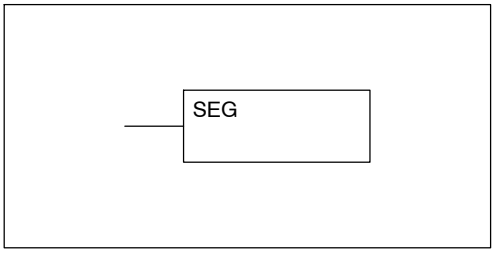
ASCII Values Valid for HTA Conversion			
Hex Value	ASCII Value	Hex Value	ASCII Value
0	30	8	38
1	31	9	39
2	32	A	41
3	33	B	42
4	34	C	43
5	35	D	44
6	36	E	45
7	37	F	46

Segment (SEG)

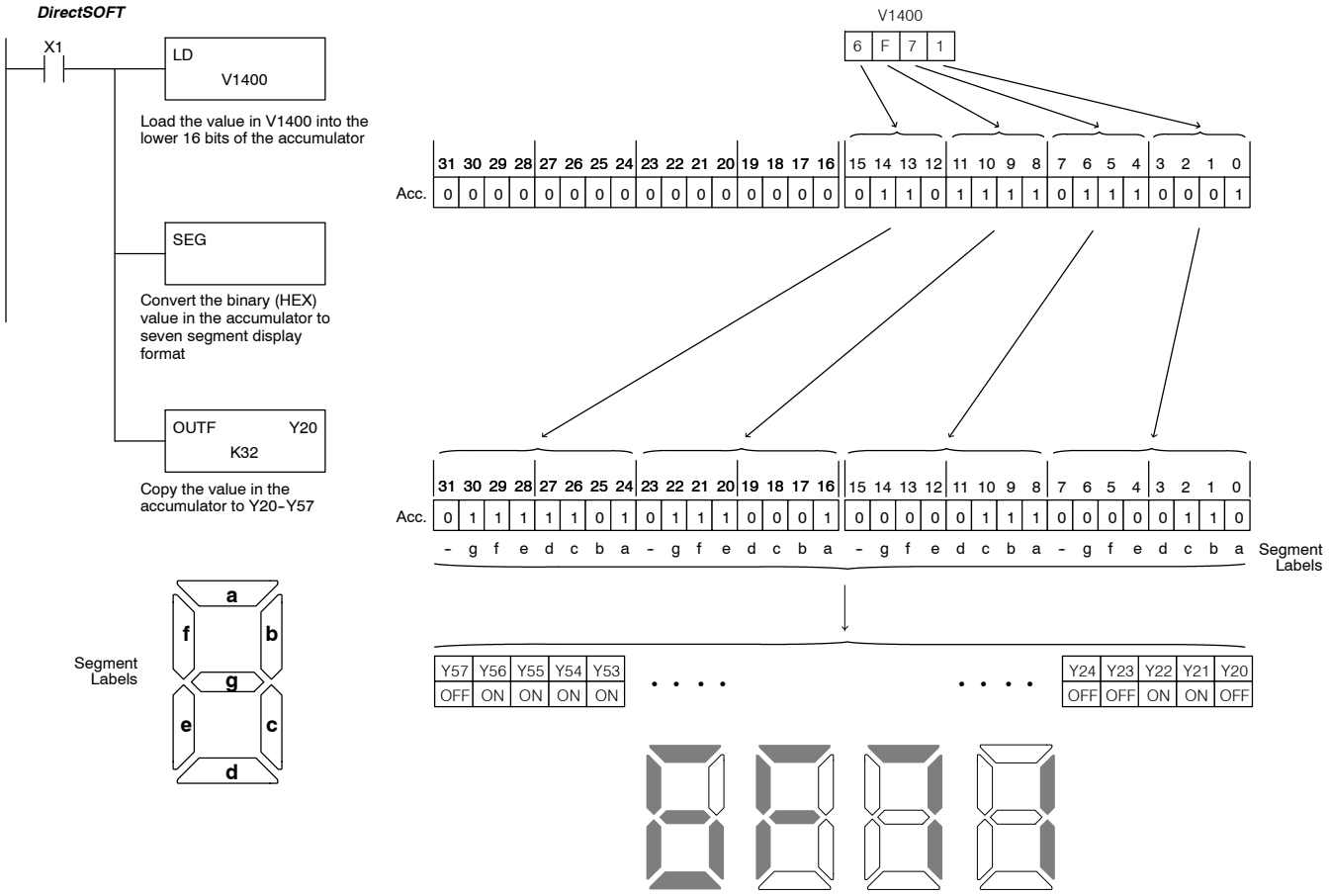
430	440	450
✓	✓	✓

DS	HPP
✓	✓

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 binary (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.

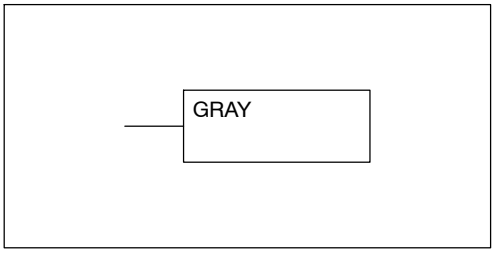


Handheld Programmer Keystrokes

STR	X(IN)	1	↵							
LD	SHFT	V	1	4	0	0	↵			
SHFT	S	E	G	↵						
OUT	SHFT	F	SHFT	Y(OUT)	2	0	K(CON)	3	2	↵

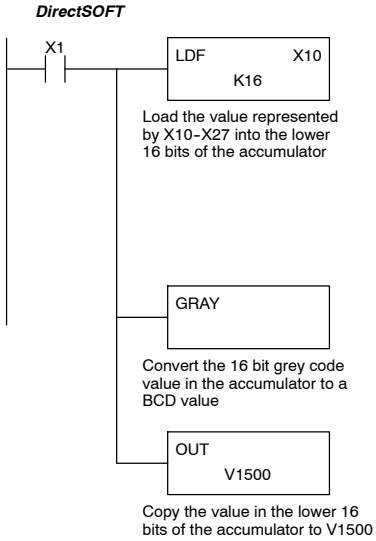
Gray Code (GRAY)

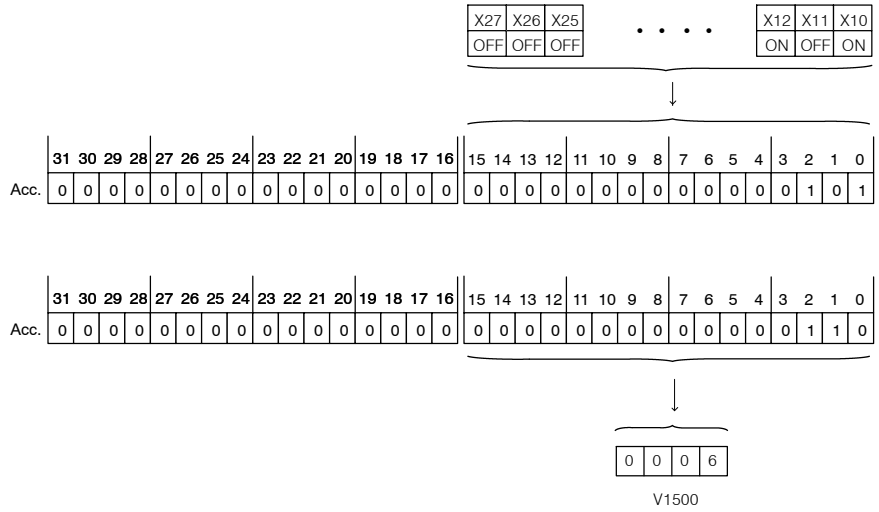
430	440	450	DS	HPP
✗	✓	✓	✓	✓



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 grey 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 V1500.





Handheld Programmer Keystrokes

STR	X(IN)	1	↵							
LD	SHFT	F	SHFT	X(IN)	1	0	K(CON)	1	6	↵
SHFT	G	R	A	Y	SHFT	↵				
OUT	V	1	5	0	0	↵				

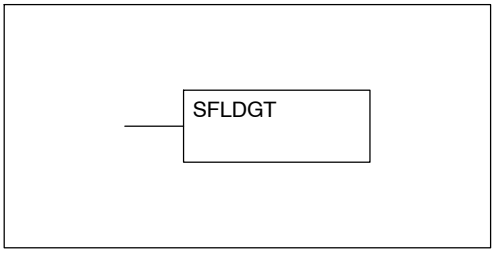
Gray Code	BCD
0000000000	0000
0000000001	0001
0000000011	0002
0000000010	0003
0000000110	0004
0000000111	0005
0000000101	0006
0000000100	0007
.	.
.	.
.	.
1000000001	1022
1000000000	1023

Shuffle Digits (SFLDGT)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

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 the digits will be shuffled to into the accumulator, numbered 1 through 8 ("1" being the least significant digit, and "8" being the most significant digit.

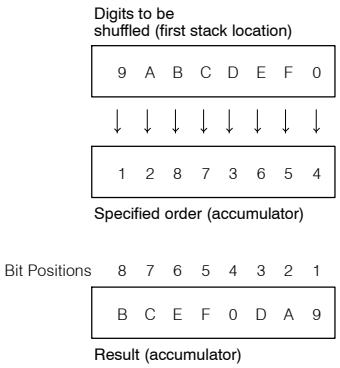
Note:— If the number used to specify the order contains a 0 or 9-F, the corresponding position will be set to 0.
See example

Note:—If the number used to specify the order contains duplicate numbers, the most significant duplicate number is valid. The result resides in the accumulator.
See example

Step 3:— Insert the SFLDGT instruction.

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 defines 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.

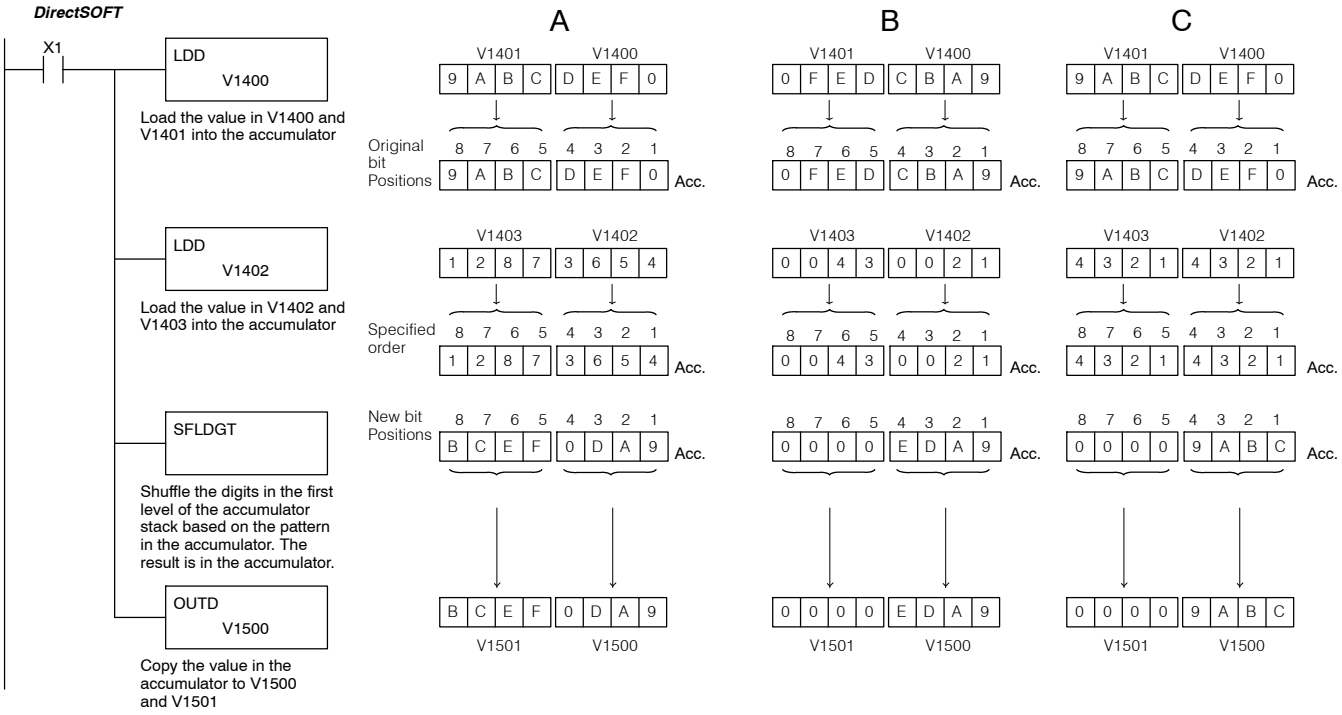


In the following example when X1 is on, The value in the first level of the accumulator stack will be reorganized in the order specified by the value in the accumulator.

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.



Handheld Programmer Keystrokes

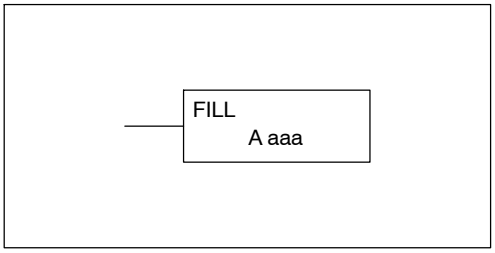
STR	X(IN)	1	↵						
LD	SHFT	D	SHFT	V	1	4	0	0	↵
LD	SHFT	D	SHFT	V	1	4	0	2	↵
SHFT	S	F	L	D	G	T	↵		
OUT	SHFT	D	SHFT	V	1	5	0	0	↵

Table Instructions

Fill (FILL)

430	440	450	DS	HPP
✓	✓	✓	✓	✓

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.

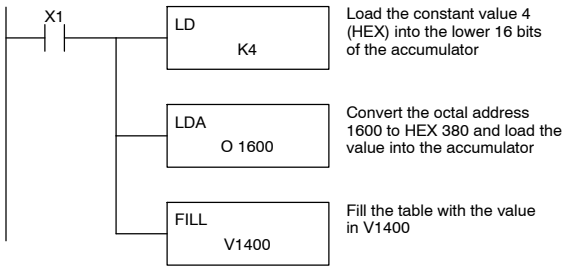
Step 3:— Insert the Fill instructions which specifies the value to fill the table with.

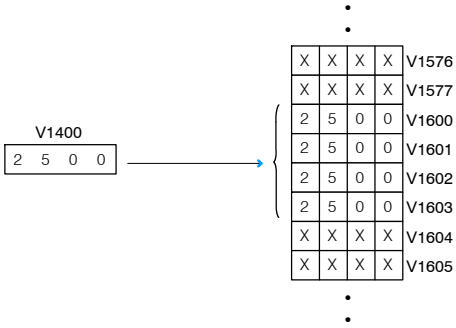
Helpful Hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-FF	0-FF	0-FF

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.

DirectSOFT





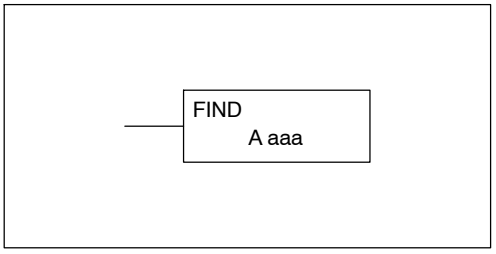
Handheld Programmer Keystrokes

STR	X(IN)	1	↵							
LD	K(CON)	4	↵							
LD	SHFT	A	OCT	1	6	0	0	↵		
SHFT	F	I	L	L	SHFT	V	1	4	0	0	↵

Find (FIND)

430	440	450	DS	HPP
✗	✓	✓	✓	✓

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 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. The result will be a HEX value.

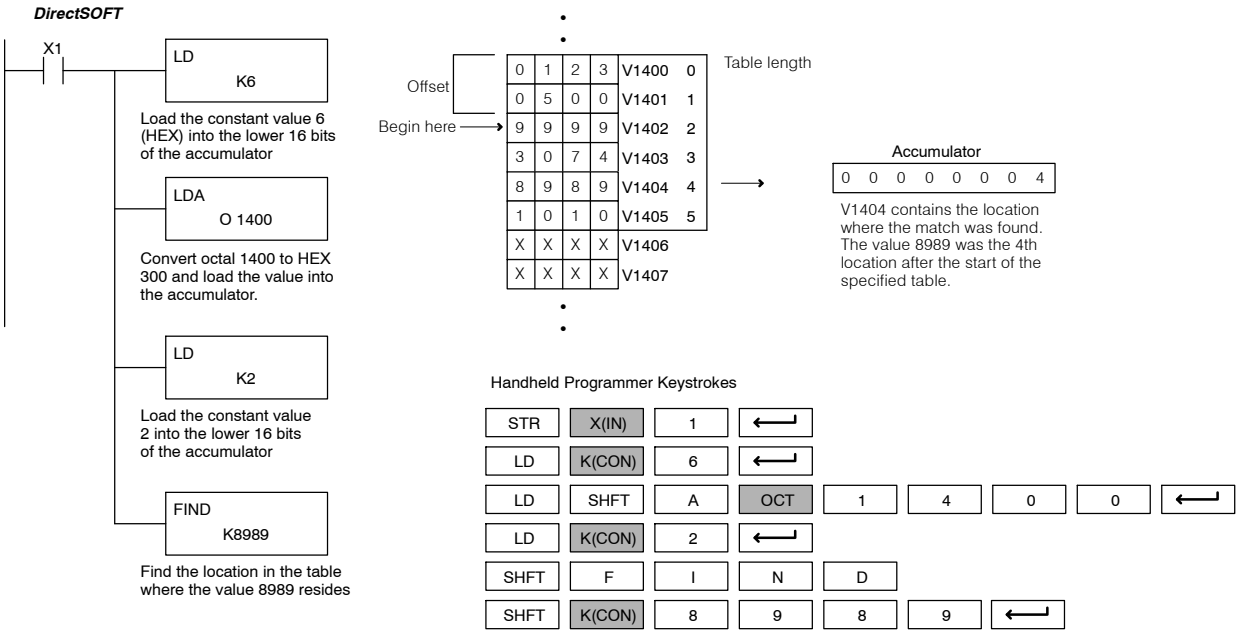
Helpful Hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		DL440 Range	DL450 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP53	ON if there is no value in the table that is equal to the search value.

NOTE: Status flags are only valid until another instruction that uses the same flags is executed. The pointer for this instruction starts at 0 and resides in the accumulator.

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the second stack location when the following Load Address and Load instruction is executed. The octal address 1400 (V1400) is the starting location for the table and is loaded into the accumulator. This value is placed in the first level of the accumulator stack when the following Load instruction is executed. The offset (K2) is loaded into the lower 16 bits of the accumulator using the Load instruction. The value to be found in the table is specified in the Find instruction. If a value is found equal to the search value, the offset (from the starting location of the table) where the value is located will reside in the accumulator.

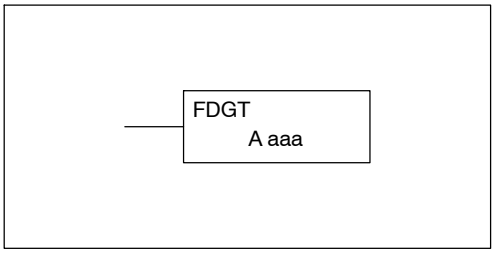


Find Greater Than (FDGT)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

The Find Greater Than instruction is used to search for the first occurrence of a value in a V-memory table that is greater than the specified value (Aaaa), which can be either a V-memory location or a 4-digit constant. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Find Greater Than function.



NOTE: This instruction does not have an offset, such as the one required for the FIND instruction.

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 instructions which specifies the greater than search value.

Results:— The offset from the starting address to the first V-memory location which contains the greater than search value is returned to the accumulator. SP53 will be set on if the value is not found and 0 will be returned in the accumulator. The result will be a HEX value.

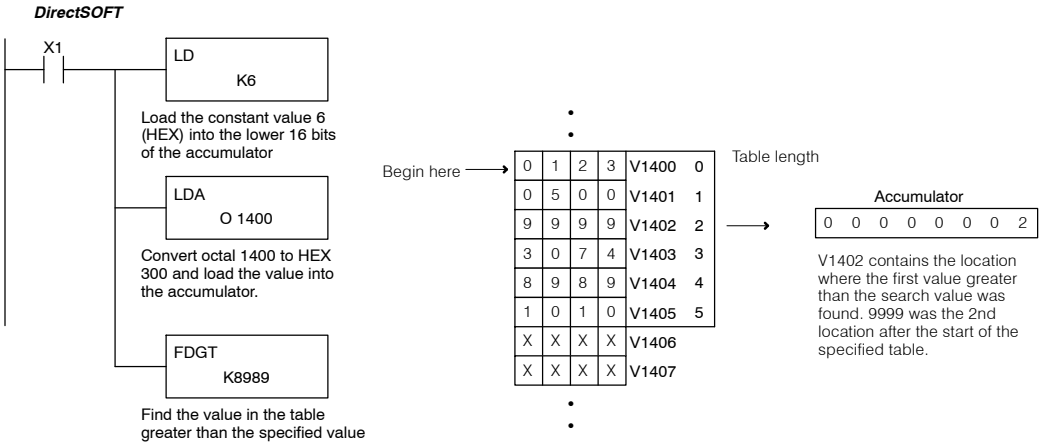
Helpful Hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		DL440 Range	DL450 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-42)
Constant	K	0-FFFF	0-FFFF

Discrete Bit Flags	Description
SP53	on if there is no value in the table that is greater than 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.



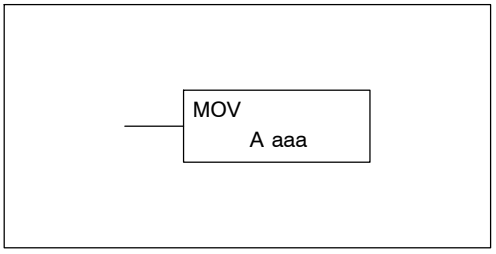
Handheld Programmer Keystrokes

STR	X(IN)	1	↵								
LD	K(CON)	6	↵								
LD	SHFT	A	OCT	1	4	0	0	↵			
SHFT	F	D	G	T	SHFT	K(CON)	8	9	8	9	↵

Move (MOV)

✗	✓	✓
430	440	450
✓		✓
DS		HPP

The Move instruction moves up to 4095 values from a V-memory table to another V-memory table the same length. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Move function.



Step 1:— Load the number of V-memory locations to be moved into the first level of the accumulator stack. This parameter must be a HEX value, 0-FFF.

Step 2:— Load the starting V-memory location for the locations to be moved into the accumulator. This parameter must be a HEX value.

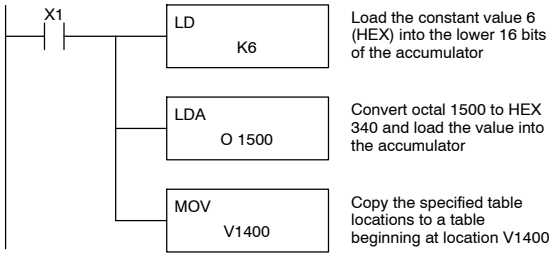
Step 3:— Insert the MOVE instruction which specifies starting V-memory location (Vaaa) for the destination table.

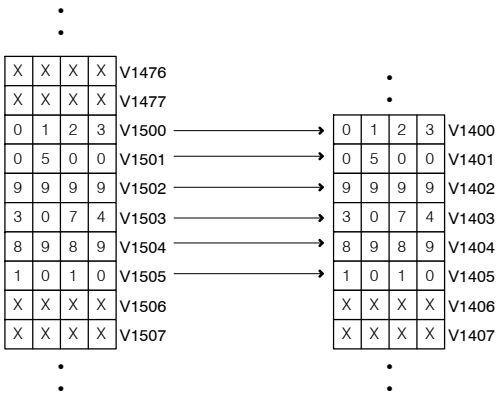
Helpful Hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		DL440 Range	DL450 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-42)
Pointer	P	All (See p. 3-41)	All (See p. 3-42)

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 1500 (V1500), the starting location for the source table is loaded into the accumulator. The destination table location (V1400) is specified in the Move instruction.

DirectSOFT





Handheld Programmer Keystrokes

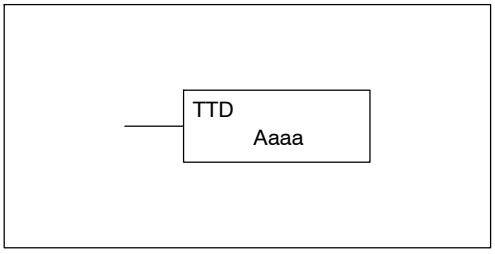
STR	X(IN)	1	↵						
LD	K(CON)	6	↵						
LD	SHFT	A	OCT	1	5	0	0	↵	
SHFT	M	O	V	SHFT	V	1	4	0	0	↵

Table to Destination (TTD)

430	440	450
	✓	✓

DS	HPP
✓	✓

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		DL440 Range	DL450 Range
	A	aaa	aaa
V-memory	V	All (See p. 3-41)	All (See p. 3-42)

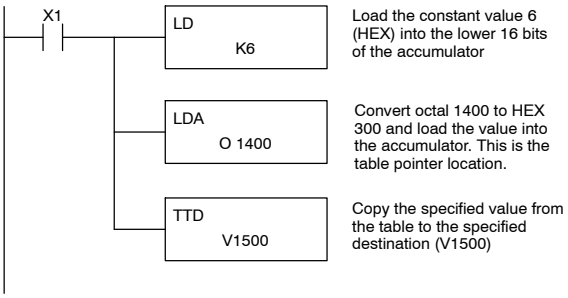
Discrete Bit Flags	Description
SP56	ON when the table pointer equals the table length.

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.

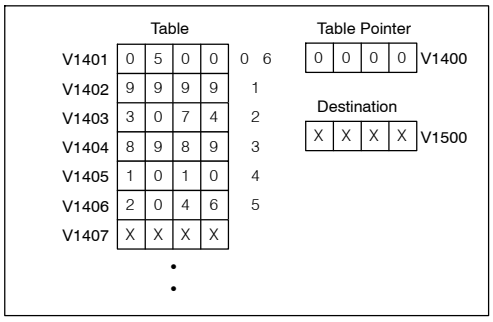
DirectSOFT



Handheld Programmer Keystrokes

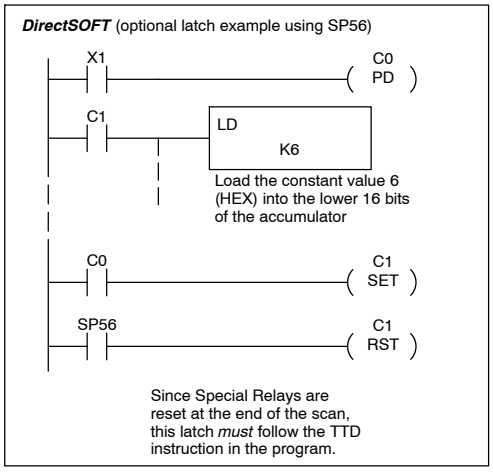
STR	X(IN)	1	↵						
LD	K(CON)	6	↵						
LD	SHFT	A	OCT	1	4	0	0	↵	
SHFT	T	T	D	SHFT	V	1	5	0	0	↵

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.



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.

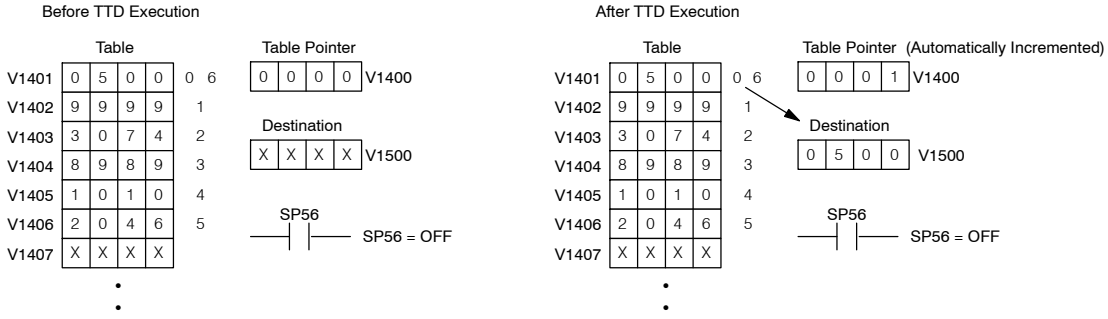
DirectSOFT (optional latch example using SP56)



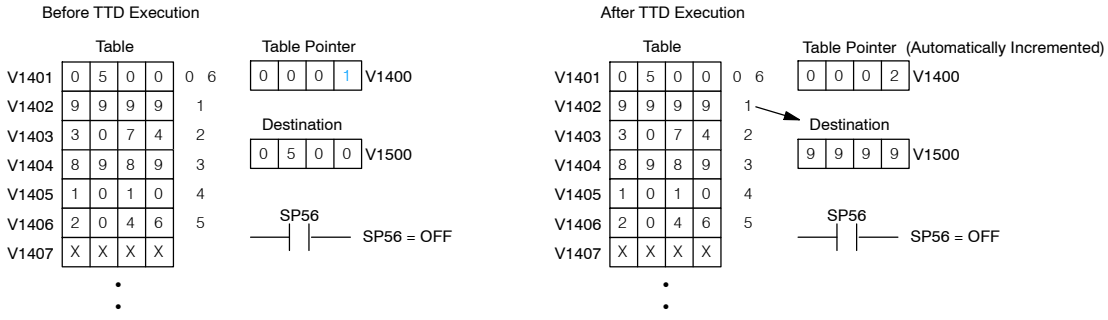
The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically cycles from 0 – 6, and then starts over at 1 instead of 0. Also, notice how SP56 is only on until the end of the scan.

Example of Execution

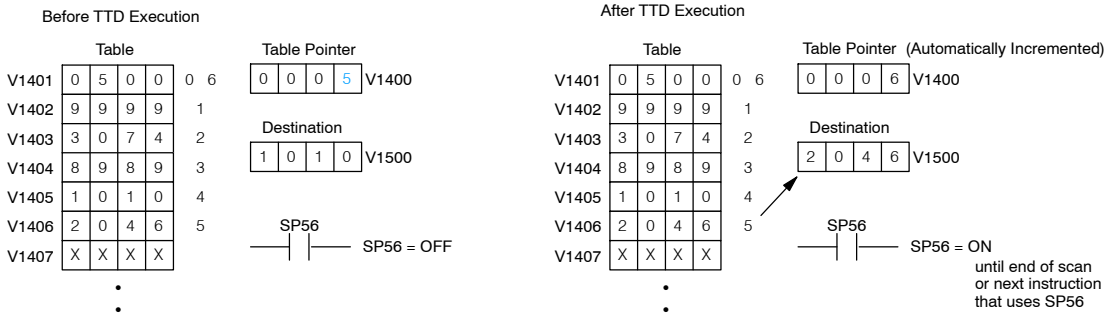
Scan N



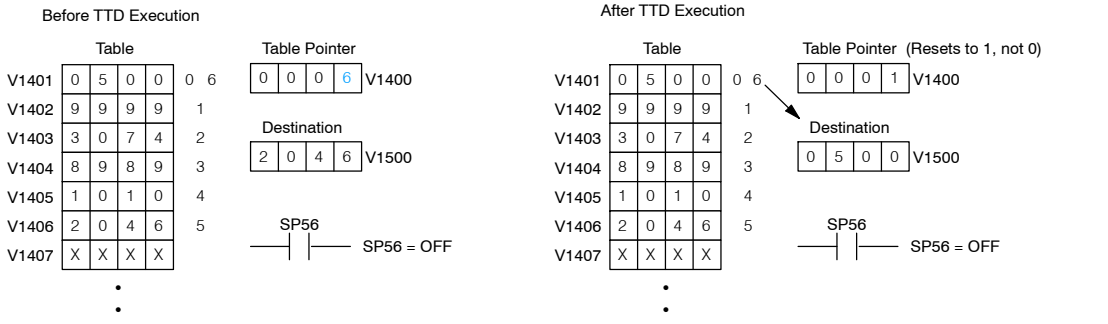
Scan N+1



Scan N+5



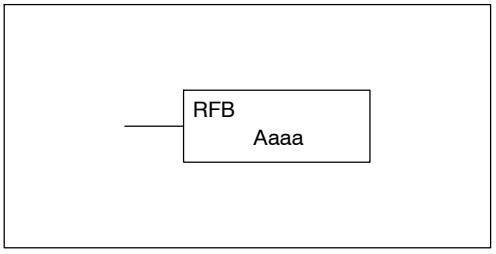
Scan N+6



Remove from Bottom (RFB)

430	440	450
✗	✓	✓

DS	HPP
✓	✓



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 instructions which specifies destination V-memory location (Vaaa).

Helpful Hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful Hint:— The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful Hint: — The pointer location should be set to the value where the table operation will begin. The special relay SP0 or a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

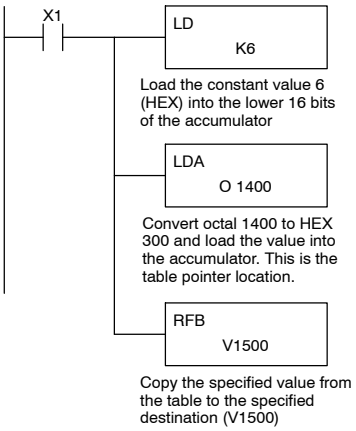
Operand Data Type		DL440 Range	DL450 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-42)

Discrete Bit Flags	Description
SP56	on when the table pointer equals 0

NOTE: Status flags (SPs) are only valid until:
— another instruction that uses the same flag is executed, or
— the end of the scan.
The pointer for this instruction can be set to start anywhere in the table. It is not set automatically. You have to load a value into the pointer somewhere in your program.

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400) is the starting location for the source table and is loaded into the accumulator. Remember, V1400 is used as the pointer location, and is not actually part of the table data source. The destination location (V1500) is specified in the Remove From Bottom. The table pointer (V1400 in this case) will be decremented by "1" after each execution of the RFB instruction.

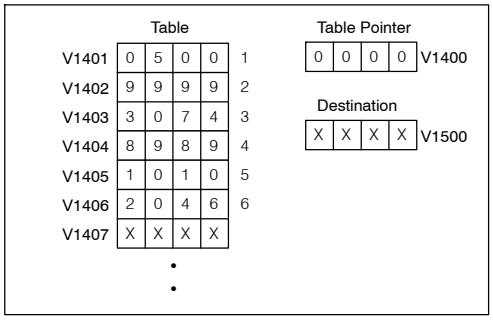
DirectSOFT



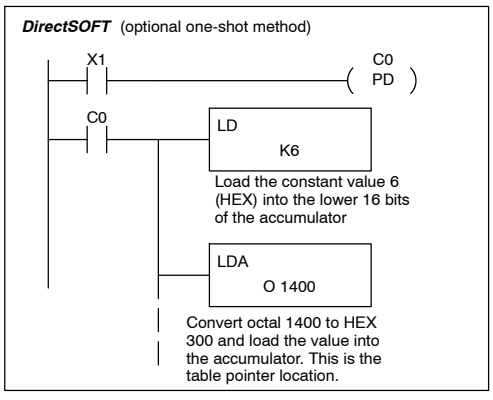
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	K(CON)	6	↵						
LD	SHFT	A	OCT	1	4	0	0	↵	
SHFT	R	F	B	SHFT	V	1	5	0	0	↵

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.



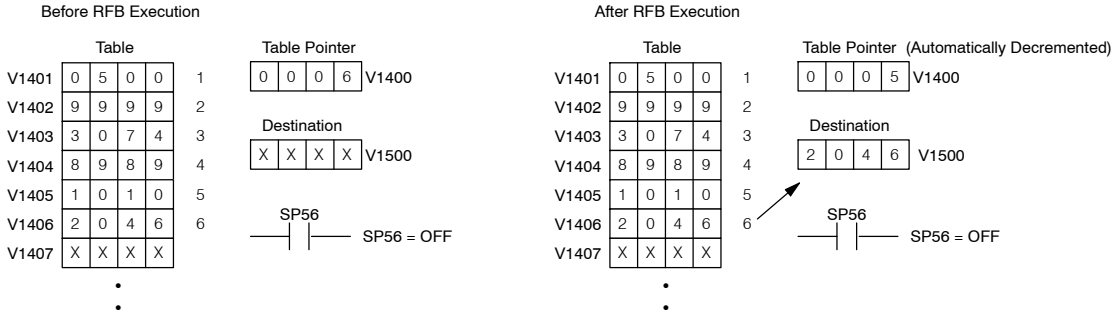
Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer decrements automatically, the table would cycle through the locations very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to remove one value each time the input contact transitions from low to high.



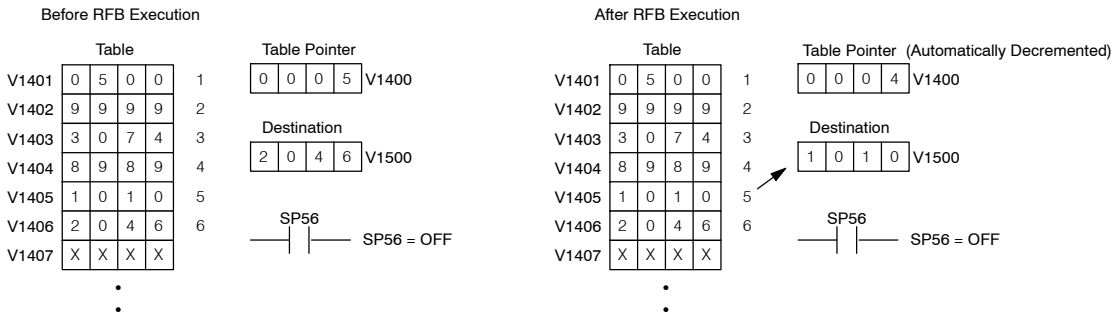
The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically decrements from 6 – 0. Also, notice how SP56 is only on until the end of the scan.

Example of Execution

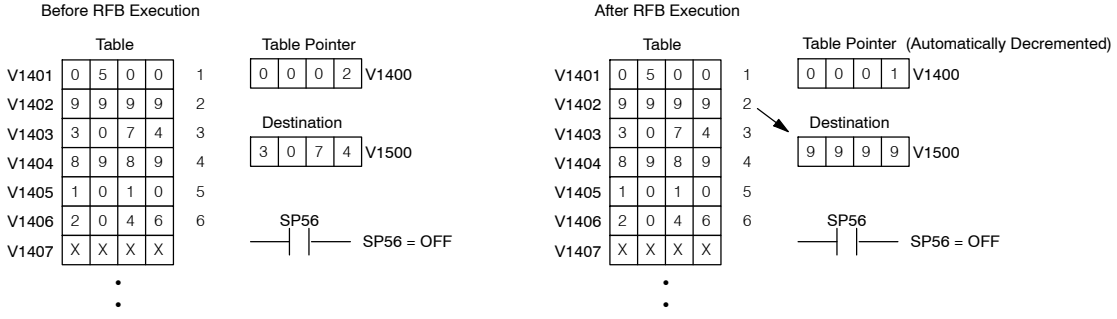
Scan N



Scan N+1



Scan N+4

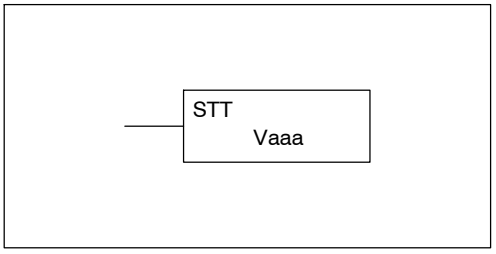


Scan N+5

Source to Table (STT)

430	440	450
✗	✓	✓

DS	HPP
✓	✓



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.

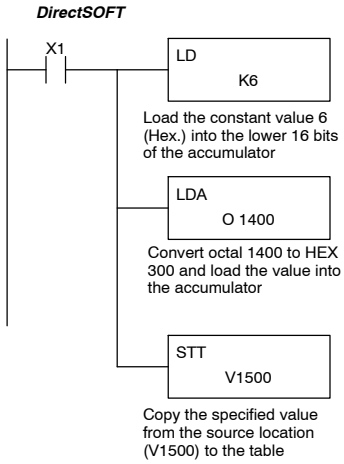
Helpful Hint: — The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the pointer should be between 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		DL440 Range	DL450 Range
		aaa	aaa
V-memory	V	All (See p. 3-41)	All (See p. 3-42)

Discrete Bit Flags	Description
SP56	on when the table pointer equals the table length

NOTE: Status flags (SPs) are only valid until:
— another instruction that uses the same flag is executed, or
— the end of the scan.
The pointer for this instruction starts at 0 and resets to 1 automatically when the table length is reached.

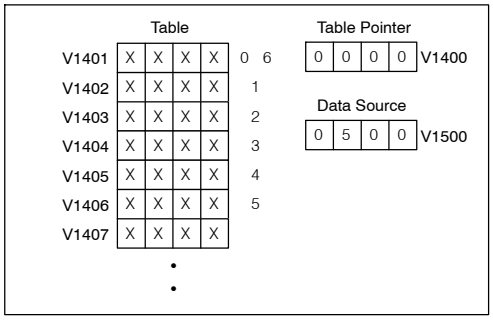
In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400), which is the starting location for the destination table and table pointer, is loaded into the accumulator. The data source location (V1500) is specified in the Source to Table instruction. The table pointer will be increased by "1" after each time the instruction is executed.



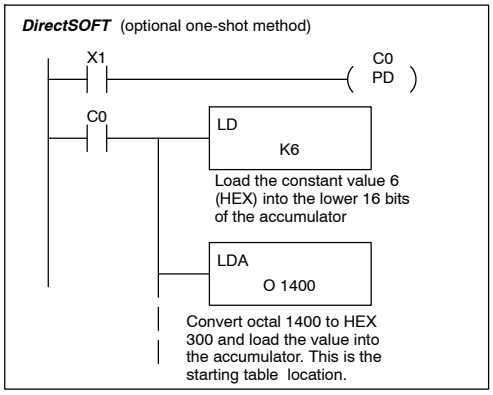
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	K(CON)	6	↵						
LD	SHFT	A	OCT	1	4	0	0	↵	
SHFT	S	T	T	SHFT	V	1	5	0	0	↵

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.



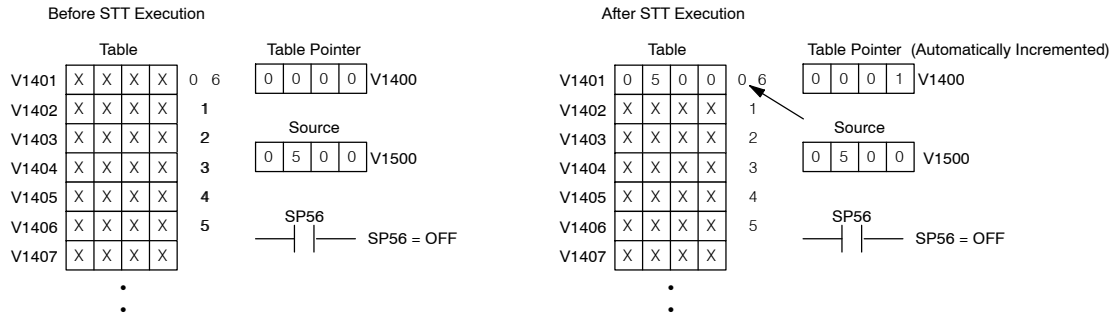
Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer increments automatically, the source data would be moved into all the table locations very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to move one value each time the input contact transitions from low to high.



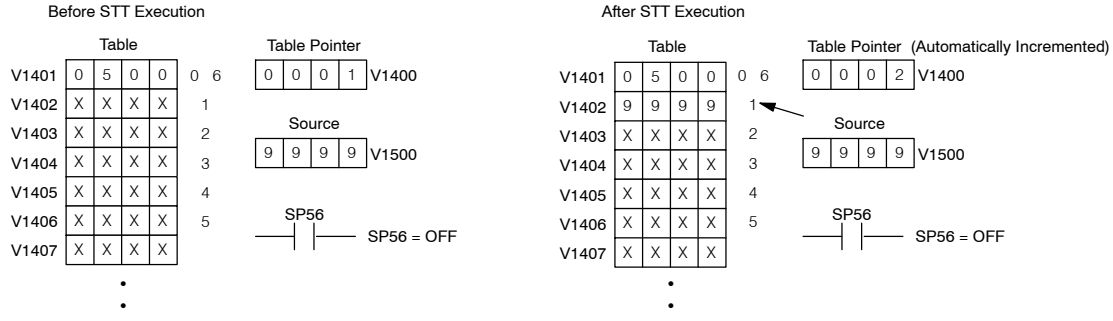
The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically cycles from 0 – 6, and then starts over at 1 instead of 0. Also, notice how SP56 is affected by the execution. Although our example does not show it, we are assuming that there is another part of the program that changes the value in V1500 (data source) prior to the execution of the STT instruction. This is not required, but it makes it easier to see how the data source is copied into the table.

Example of Execution

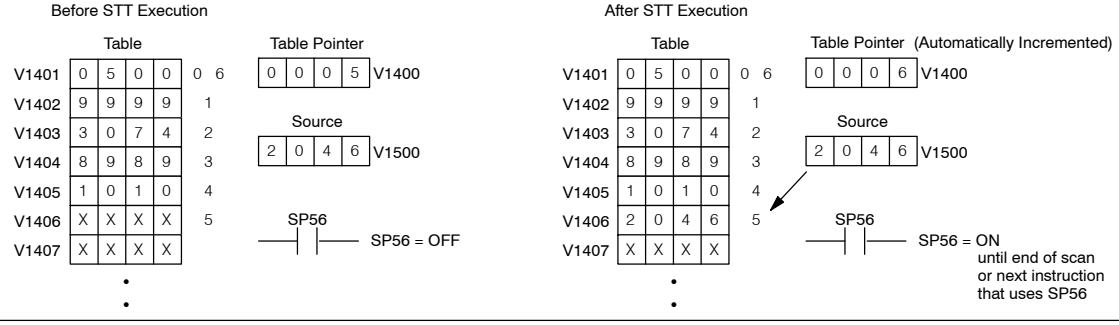
Scan N



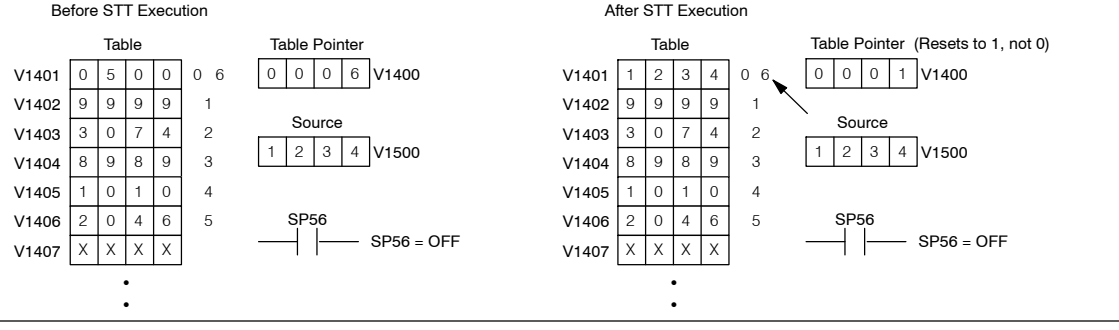
Scan N+1



Scan N+5



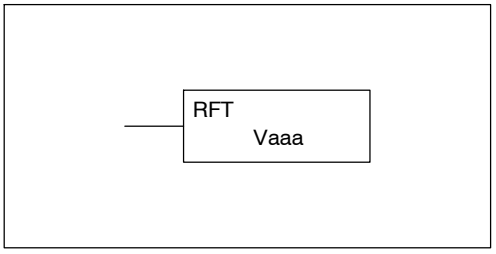
Scan N+6



Remove from Table (RFT)

430	440	450
✗	✓	✓

DS	HPP
✓	✓



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 instruction which specifies destination V-memory location (Vaaa). The value will be moved to this location.

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 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.

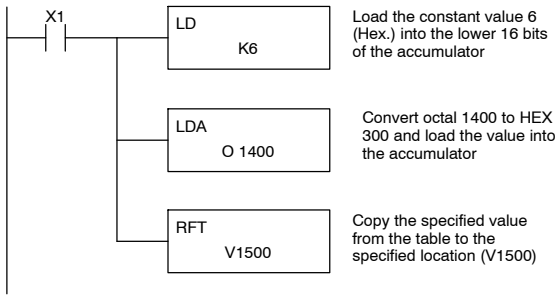
Operand Data Type		DL440 Range	DL450 Range
		aaa	aaa
V-memory	V	All (See p. 3-41)	All (See p. 3-42)

Discrete Bit Flags	Description
SP56	on when the table counter equals 0

NOTE: Status flags (SPs) are only valid until:
— another instruction that uses the same flag is executed, or
— the end of the scan.
The pointer for this instruction can be set to start anywhere in the table. It is not set automatically. You have to load a value into the pointer somewhere in your program.

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400) is the starting location for the source table and is loaded into the accumulator. The destination location (V1500) is specified in the Remove from Table instruction. The table counter will be decreased by "1" after the instruction is executed.

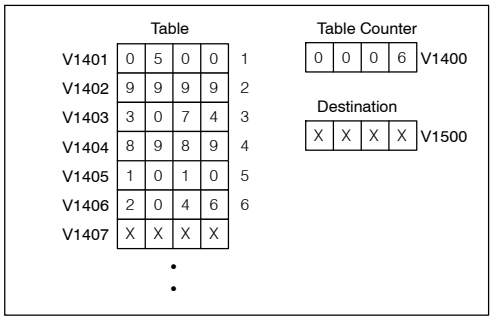
DirectSOFT



Handheld Programmer Keystrokes

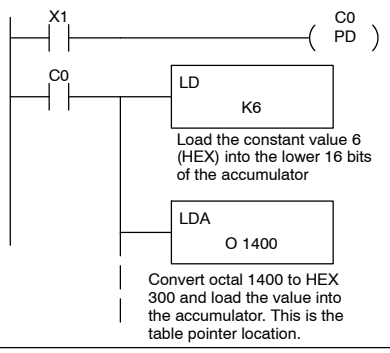
STR	X(IN)	1	↵							
LD	K(CON)	6	↵							
LD	SHFT	A	OCT	1	4	0	0	↵		
SHFT	R	F	T	SHFT	V	1	5	0	0	↵

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.



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.

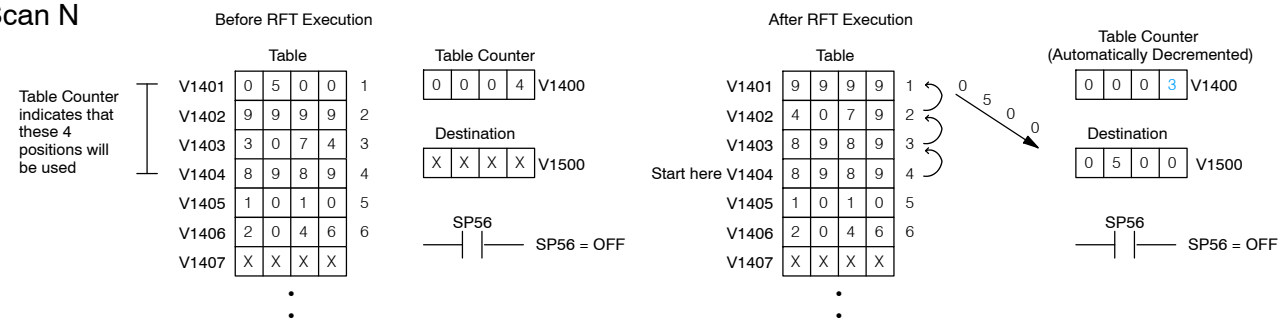
DirectSOFT (optional one-shot method)



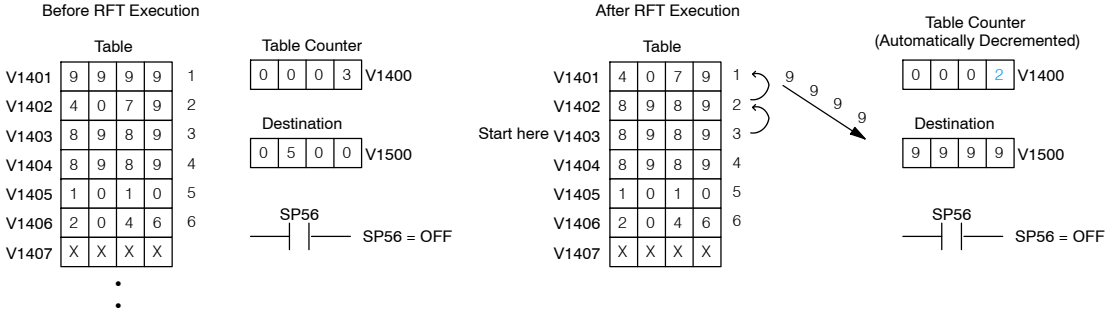
The following diagram shows the scan-by-scan results of the execution for our example program. In our example we're showing the table counter set to 4 initially. (Remember, you can set the table counter to any value that is within the range of the table.) The table counter automatically decrements from 4–0 as the instruction is executed. Notice how the last two table positions, 5 and 6, are not moved up through the table. Also, notice how SP56, which comes on when the table counter is zero, is only on until the end of the scan.

Example of Execution

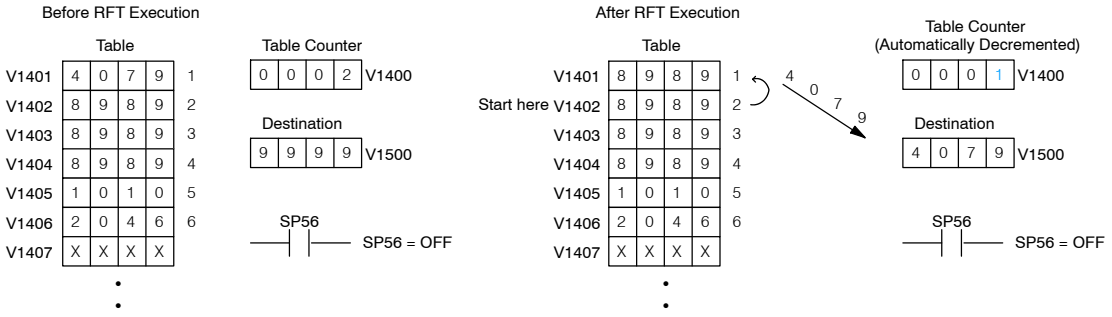
Scan N



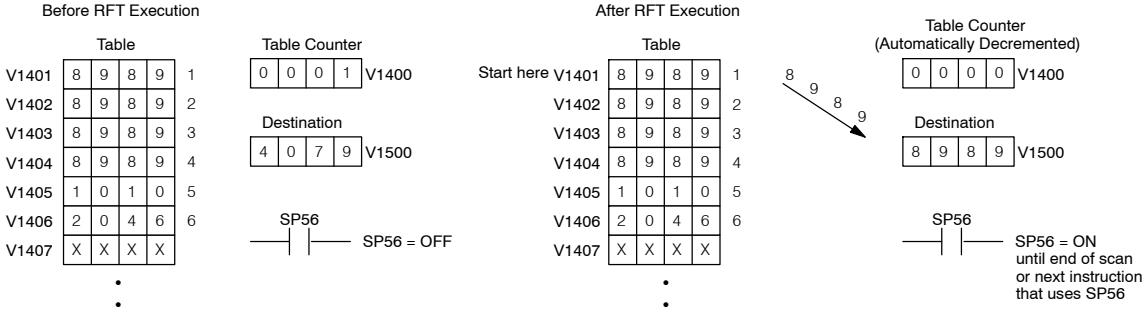
Scan N+1



Scan N+2



Scan N+3

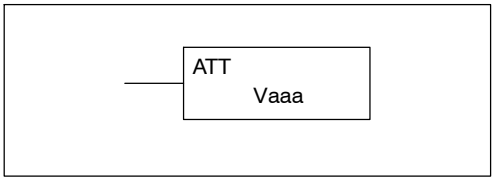


Add to Top (ATT)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

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). The value will be moved from this location.

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.

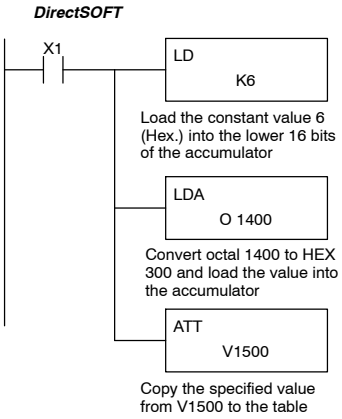
Helpful Hint: — The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the table counter should be between 1 and 6. If the value is outside 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		DL440 Range	DL450 Range
		aaa	aaa
V-memory	V	All (See p. 3-41)	All (See p. 3-42)

Discrete Bit Flags	Description
SP56	on when the table counter is 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 have to load a value into the pointer somewhere in your program.

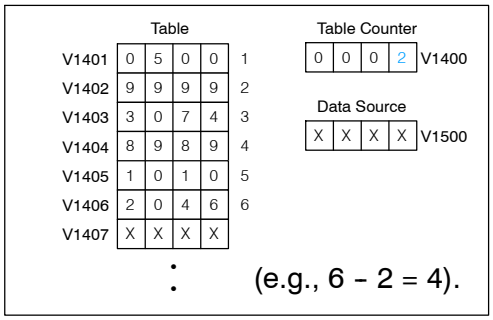
In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400), which is the starting location for the destination table and table counter, is loaded into the accumulator. The source location (V1500) is specified in the Add to Top instruction. The table counter will be increased by "1" after the instruction is executed.



Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
LD	K(CON)	6	↵						
LD	SHFT	A	OCT	1	4	0	0	↵	
SHFT	A	T	T	SHFT	V	1	5	0	0	↵

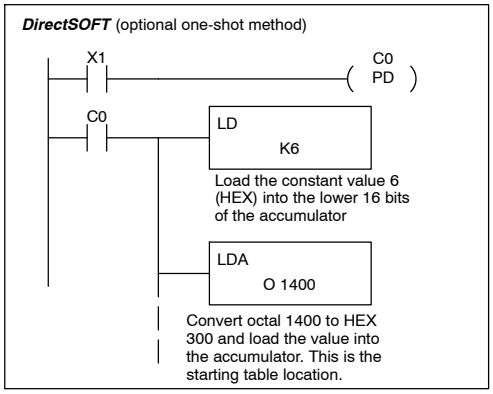
For the ATT instruction, the table counter determines the number of additions that can be made before the instruction will stop executing. So, it is helpful to understand how the system uses this counter to control the execution.



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

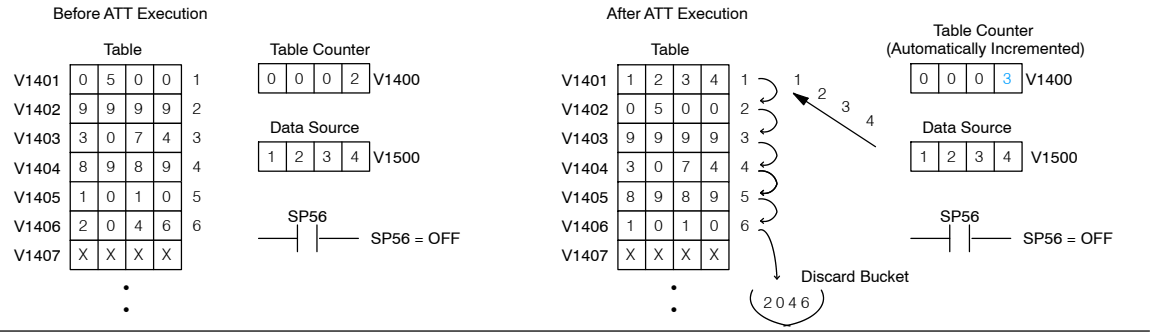
Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the table counter increments automatically, the data would be moved into the table very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to add one value each time the input contact transitions from low to high.



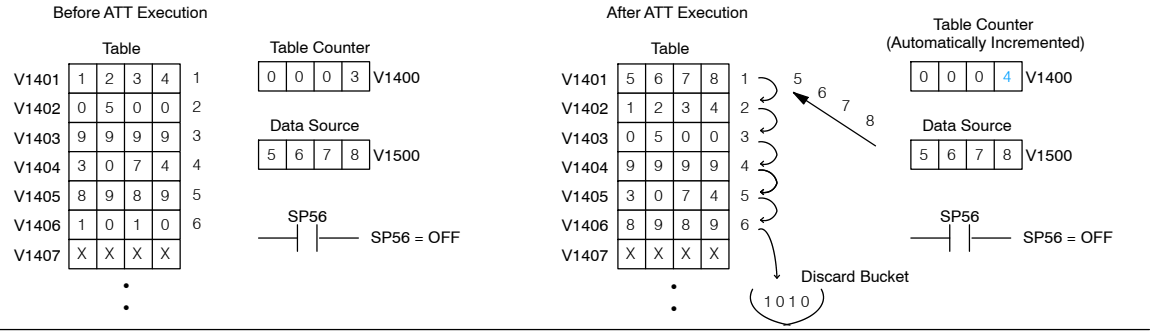
The following diagram shows the scan-by-scan results of the execution for our example program. The table counter is set to 2 initially, and it will automatically increment from 2 – 6 as the instruction is executed. Notice how SP56 comes on when the table counter is 6, which is equal to the table length. Plus, although our example does not show it, we are assuming that there is another part of the program that changes the value in V1500 (data source) prior to the execution of the ATT instruction.

Example of Execution

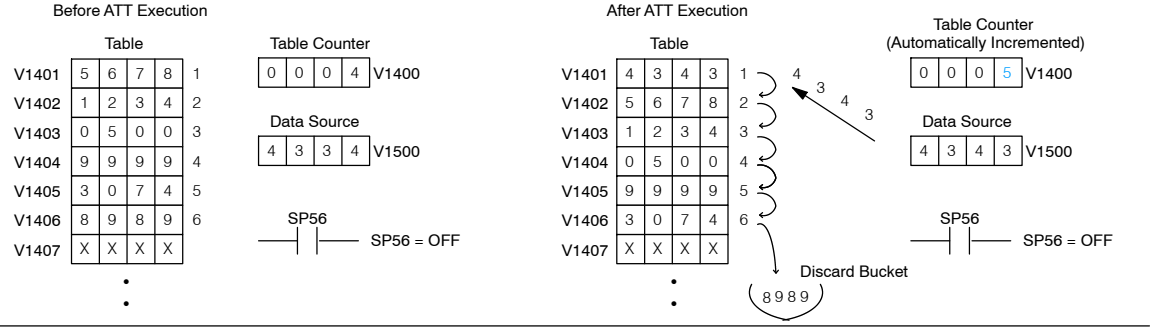
Scan N



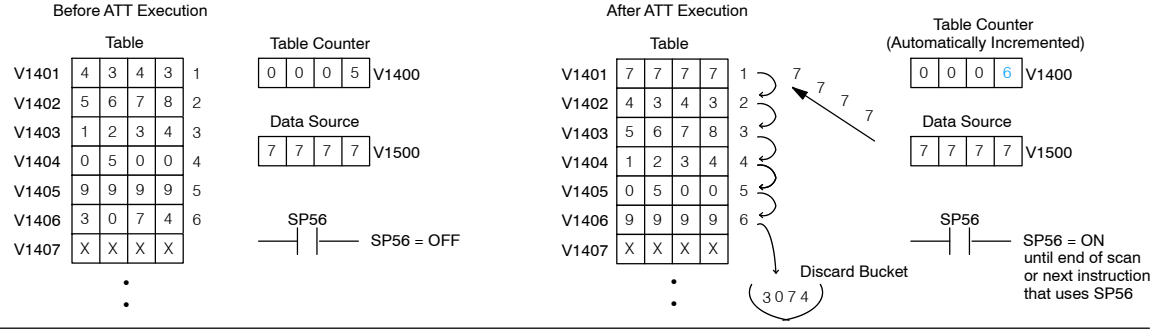
Scan N+1



Scan N+2



Scan N+3



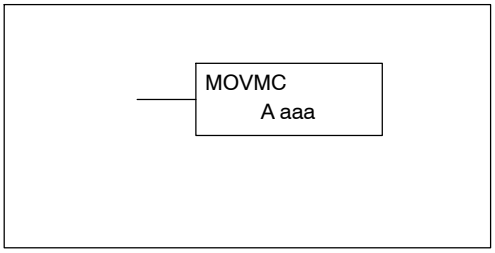
Move Memory Cartridge/ Load Label (MOVMC/LDLBL)

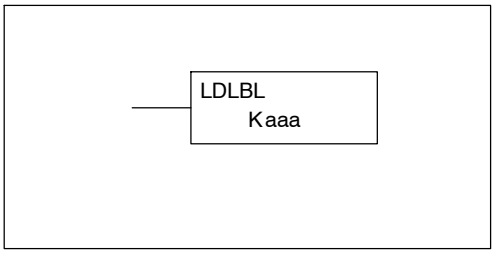
430	440	450
✗	✓	✓

DS	HPP
✓	✓

The Move Memory Cartridge instruction is used to copy data between V-memory and program ladder memory. The Load Label instruction is *only* used with the MOVMC instruction when copying data *from* program ladder memory *to* V-memory.

To copy data between V-memory and program ladder memory, the function parameters are loaded into the first two levels of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Move Memory Cartridge and Load Label functions.





Step 1:— Load the number of words (255 maximum) to be copied into the second level of the accumulator stack. This must be a hex value, 0 to FF.

Step 2:— Load the offset for the data label area (in HEX) in the program ladder memory and the beginning of the V-memory block into the first level of the accumulator stack.

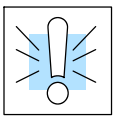
Step 3:— Load the *source data label* (LDLBL Kaaa) into the accumulator when copying data from ladder memory to V-memory. Load the *source address* into the accumulator when copying data from V-memory to ladder memory. The value will be copied from this location. If the source address is a V-memory location, the value must be entered in HEX.

Step 4:— Insert the MOVMC instruction which specifies destination (Aaaa). The value will be copied to this location.

Operand Data Type		DL440 Range	DL450 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-42)
Constant	K	1-FFFF	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, unknown data values will be transferred into the destination table.

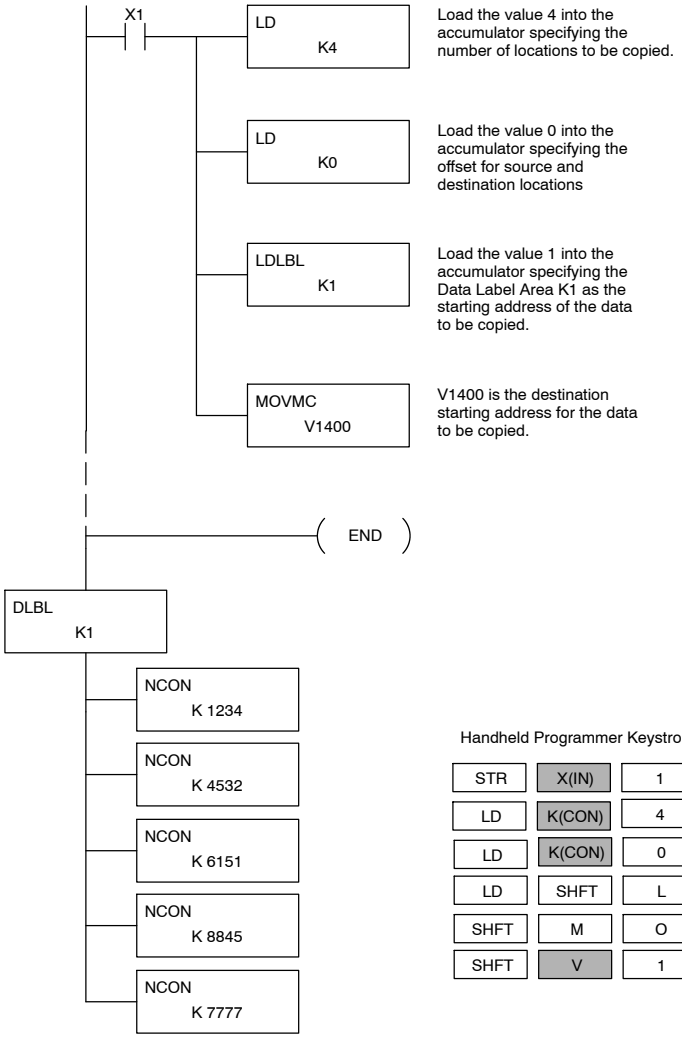
Copy Data From a Data Label Area to V-memory

430	440	450
✗	✓	✓

DS	HPP
✓	✓

In the following example, data is copied from a Data Label Area to V-memory. When X1 is on, the constant value (K4) is loaded into the accumulator using the Load instruction. This value specifies the length of the destination table and is placed in the second stack location after the next Load and Load Label (LDLBL) instructions are executed. The constant value (K0) is loaded into the accumulator using the Load instruction. This value specifies the offset for the source *and* the destination table, and is placed in the first stack location after the LDLBL instruction is executed. The source address where data is being copied from is loaded into the accumulator using the LDLBL instruction. The MOVMC instruction specifies the destination table starting location and executes the copying of data from the source Data Label Area to V-memory.

DirectSOFT



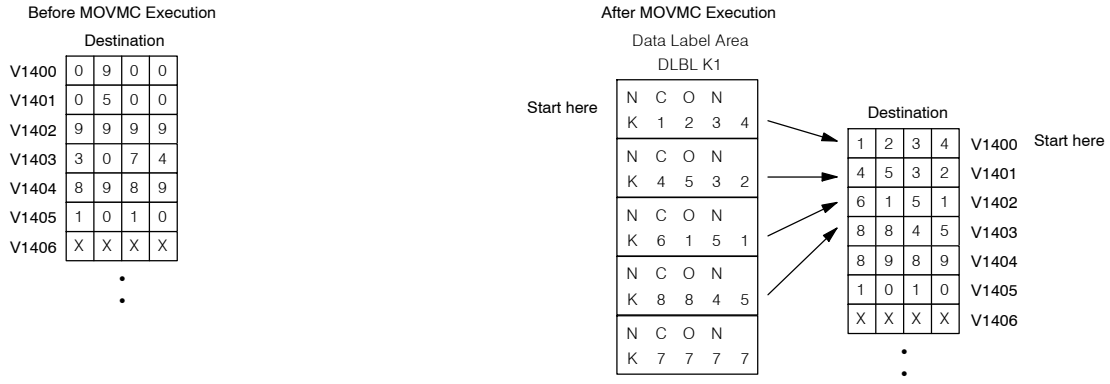
Handheld Programmer Keystrokes (for MOVMC portion only)

STR	X(IN)	1	↵					
LD	K(CON)	4	↵					
LD	K(CON)	0	↵					
LD	SHFT	L	B	L	SHFT	K(CON)	1	↵
SHFT	M	O	V	M	C			
SHFT	V	1	4	0	0	↵		

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.

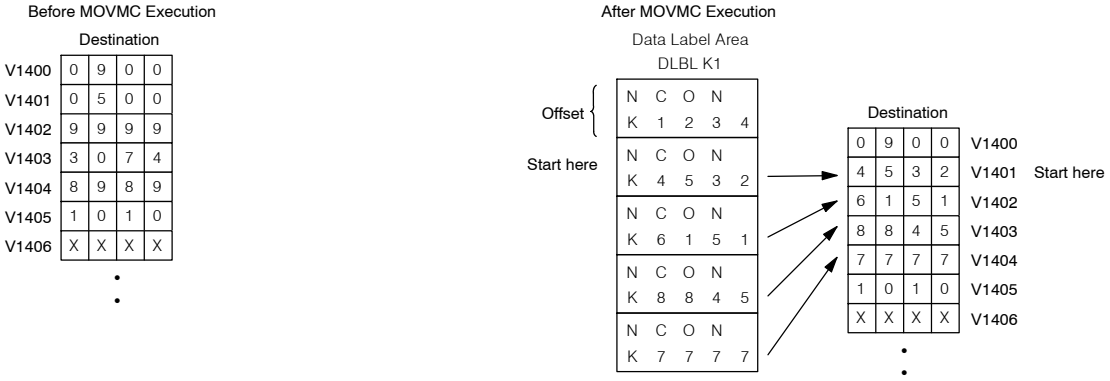
The following diagram shows the result of our example. The offset is equal to zero and four words will be copied into the Data Label area.

Example of Execution
Offset = 0, move 4 words

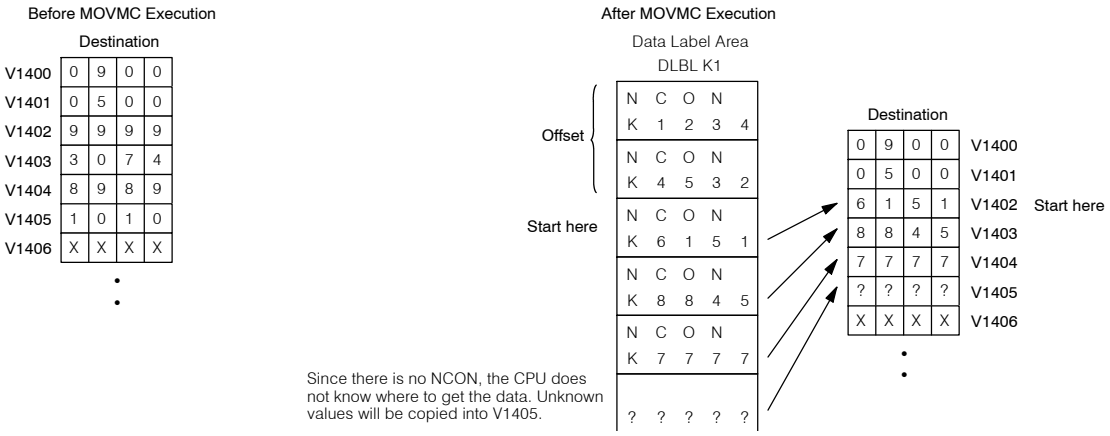


The example is fairly straightforward when an offset of zero is used. However, it is also helpful for you to understand the results that would have been obtained if different offset values (1 and 2) were used. Notice how the offset is used for both the data label (source) *and* the destination table. 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



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.

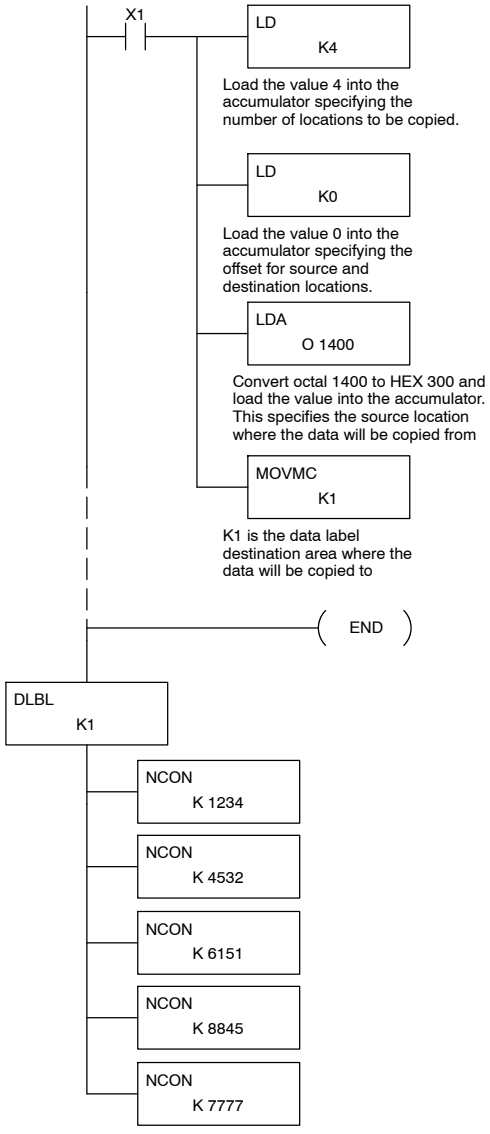
Copy Data From V-memory to a Data Label Area

430	440	450
✗	✓	✗

DS	HPP
✓	✓

NOTE: You must use a RAM cartridge for this example to work.
In this example, data is copied from V -memory to a data label area. When X1 is on, the constant value (K4) is loaded into the accumulator using the Load instruction. This value specifies the length of the destination table and is placed in the second stack location after the next Load and Load Address instructions are executed. The constant value (K0) is loaded into the accumulator using the Load instruction. This value specifies the offset for the source *and* destination table, and is placed in the first stack location after the Load Address instruction is executed. The source address data is being copied from is loaded into the accumulator using the Load Address instruction. The MOVMC instruction specifies the destination starting location and executes the copying of data from V-memory to the data label area.

DirectSOFT



Handheld Programmer Keystrokes (for MOVMC portion only)

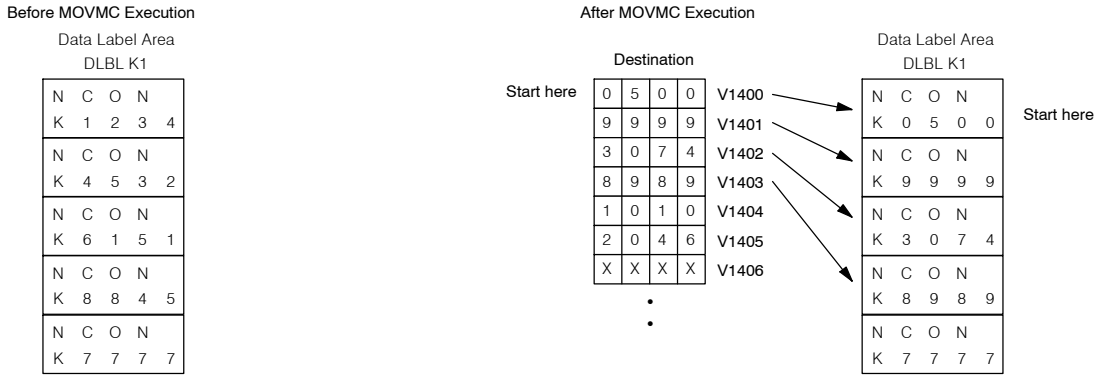
STR	X(IN)	1	↵					
LD	K(CON)	4	↵					
LD	K(CON)	0	↵					
LD	SHFT	A	OCT	1	4	0	0	↵
SHFT	M	O	V	M	C	SHFT	K(CON)	1

Note: This instruction works only with the RAM cartridge. It does not work with the EEPROM.

WARNING: The offset for this usage of the instruction starts at 0. If the offset (or the specified data table range) is large enough to cause data to be copied from V-memory to beyond the end of the DLBL area, then anything after the specified DLBL area will be replaced with invalid instructions.

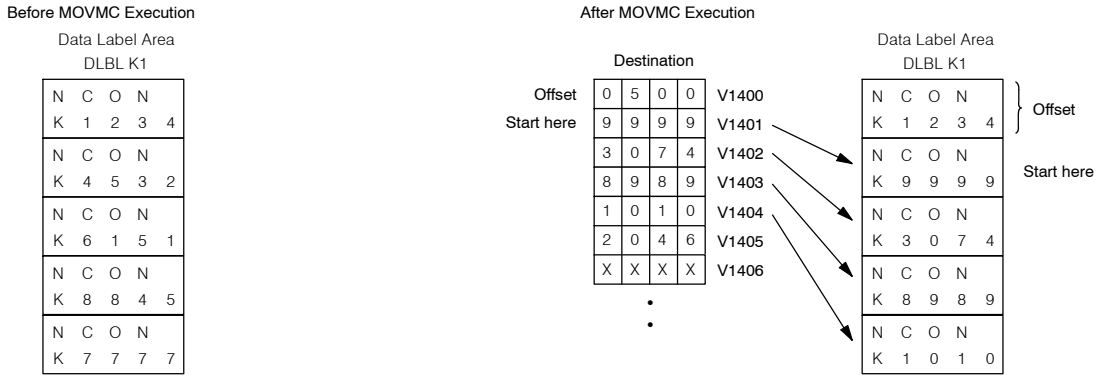
The following diagram shows the result of our example. The offset is equal to zero and four words will be copied into the Data Label area.

Example of Execution
Offset = 0, move 4 words

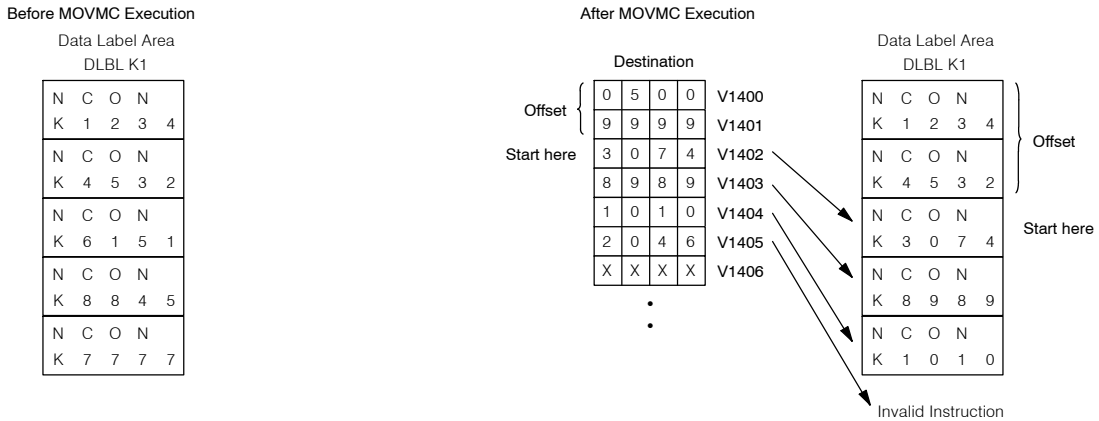


The example is fairly straightforward when an offset of zero is used. However, it is also helpful for you to understand the results that would have been obtained if different offset values (1 and 2) were used. Notice how the offset is used for both the V-memory data table (source) *and* the Data Label area. Also, notice how an improper offset (two in this case) can result in invalid instructions being written over any instructions that follow the Data Label.

Offset = 1, move 4 words



Offset = 2, move 4 words

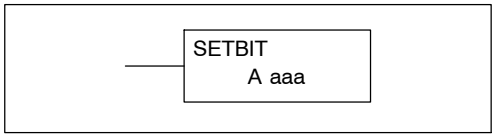


Set Bit (SETBIT)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

The Set Bit instruction sets a single bit to one within a range of V-memory locations.

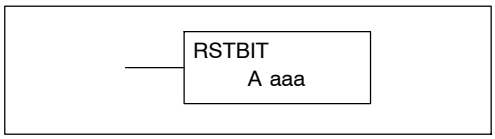


Reset Bit (RSTBIT)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

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".

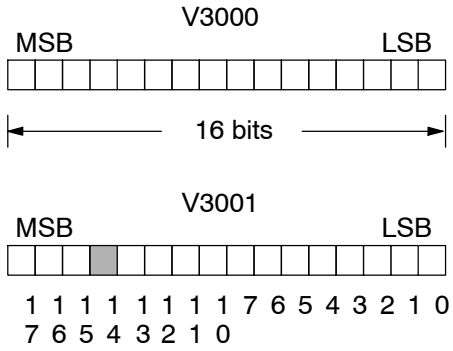
Helpful hint: — Remember that each V-memory location contains 16 bits. So, the bits of the first word of the table are numbered from 0 to 17 octal. For example, if the table length is six words, then 6 words = (6 x 16) bits, = 96 bits (decimal), or 140 octal. The permissible range of bit reference numbers would be 0 to 137 octal. Flag 53 will be set if the bit specified is outside the range of the table.

Operand Data Type		DL450 Range
		aaa
V-memory	V	All (See p. 3-42)
Octal Address	O	0-7777

Discrete Bit Flags	Description
SP53	on when the bit number which is referenced in the Set Bit or Reset Bit exceeds the range of the table

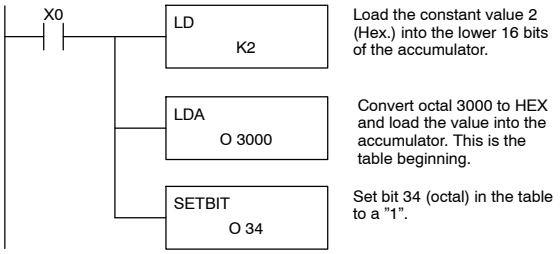
NOTE: Status flags are only valid until:
— the end of the scan
— or another instruction that uses the same flag is executed.

For example, suppose we have a table starting at V3000 that is two words long, as shown to the right. Each word in the table contains 16 bits, or 0 to 17 in octal. To set bit 12 in the second word, we use its octal reference (bit 14). Then we compute the bit's octal address from the start of the table, so 17 + 14 = 34 octal. The following program shows how to set the bit as shown to a "1".



In this ladder example, we will use input X0 to trigger the Set Bit operation. First, we will load the table length (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 beginning.

DirectSOFT



Handheld Programmer Keystrokes

STR	X(IN)	0	↵					
LD	K(CON)	2	↵					
LD	SHFT	A	OCT	3	0	0	0	↵
SET	SHFT	B	I	T	OCT	3	4	↵

Table Shift Left (TSHFL)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

The Table Shift Left instruction shifts all the bits in a V-memory table to the left, the specified number of bit positions.

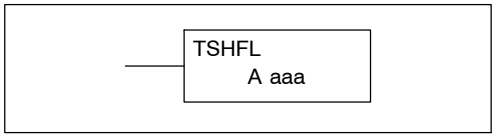
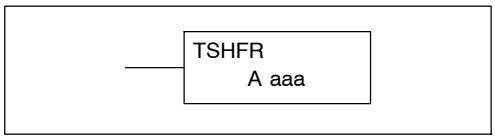


Table Shift Right (TSHFR)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

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.

Table Shift Left

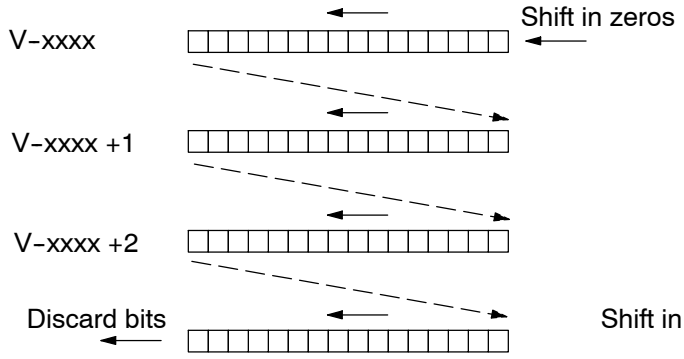
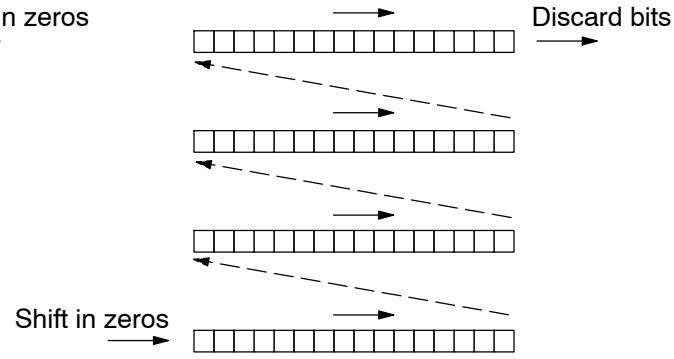


Table Shift Right



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.

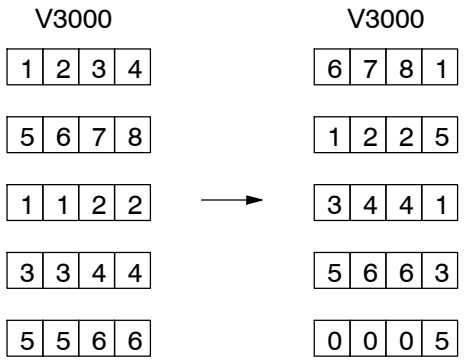
Helpful hint: — Remember that each V-memory location contains 16 bits. So, the bits of the first word of the table are numbered from 0 to 17 octal. If you want to shift the entire table by 20 bits, that is 24 octal. Flag 53 will be set if the number of bits to be shifted is larger than the total bits contained within the table. Flag 67 will be set if the last bit shifted (just before it is discarded) is a "1".

Operand Data Type		DL450 Range
		aaa
V-memory	V	All (See p. 3-42)

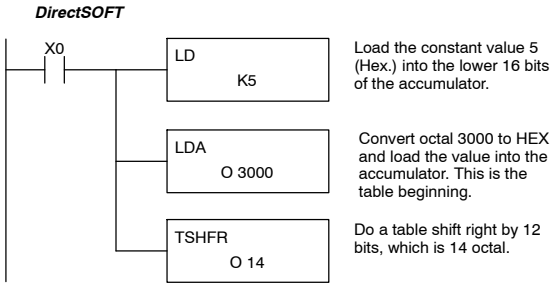
Discrete Bit Flags	Description
SP53	on when the number of bits to be shifted is larger than the total bits contained within the table
SP67	on when the last bit shifted (just before it is discarded) is a "1"

NOTE: Status flags are only valid until:
— the end of the scan
— or another instruction that uses the same flag is executed.

The example table to the right contains BCD data as shown (for demonstration purposes). Suppose we want to do a table shift right by 3 BCD digits (12 bits). Converting to octal, 12 bits is 14 octal. Using the Table Shift Right instruction and specifying a shift by octal 14, we have the resulting table shown at the far right. Notice that the 2-3-4 sequence has been discarded, and the 0-0-0 sequence has been shifted in at the bottom.



The following ladder example assumes the data at V3000 to V3004 already exists as shown above. We will use input X0 to trigger the Table Shift Right operation. First, we will load the table length (5 words) into the accumulator stack. Next, we load the starting address into the accumulator. Since V3000 is an octal number we have to convert it to hex by using the LDA command. Finally, we use the Table Shift Right instruction and specify the number of bits to be shifted (12 decimal), which is 14 octal.



Handheld Programmer Keystrokes

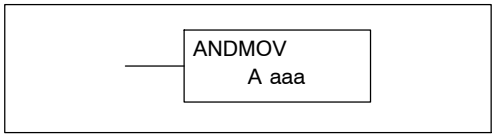
STR	X(IN)	0	↵					
LD	K(CON)	5	↵					
LD	SHFT	A	OCT	3	0	0	0	↵
SHFT	T	S	H	F	R	OCT	1	4	↵

AND Move (ANDMOV)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

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.

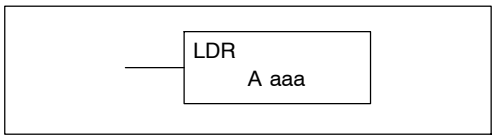


OR Move (ORMOV)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

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.

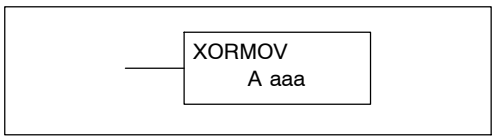


Exclusive OR Move (XORMOV)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

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.

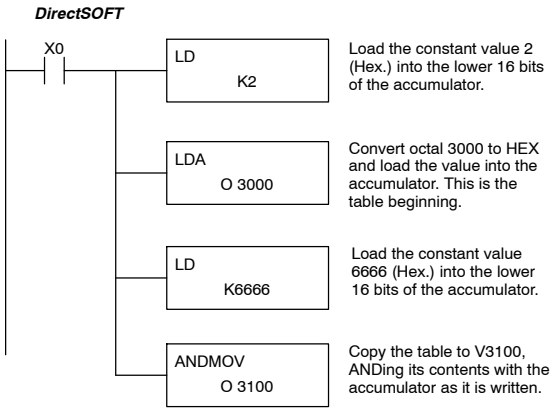
Step 3: — Load the BCD/hex bit pattern into the accumulator which will be logically combined with the table contents as they are copied.

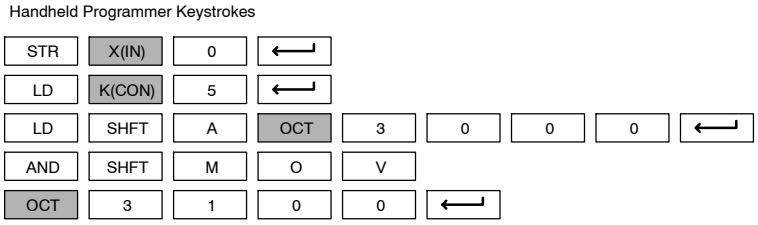
Step 4: —Insert the AND Move, OR Move, or XOR Move instruction. This specifies the starting location of the copy of the original table. This new table will automatically be the same length as the original table.

Operand Data Type		DL450 Range
		aaa
V-memory	V	All (See p. 3-42)

The example table to the right contains BCD data as shown (for demonstration purposes). Suppose we want to move a table of two words at V3000 and AND it with K6666. The copy of the table at V3100 shows the result of the AND operation for each word.

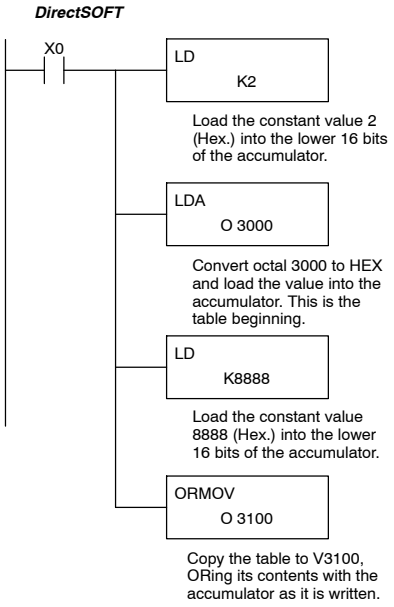
The program on the next page performs the ANDMOV operation example above. It assumes that the data in the table at V3000 – V3001 already exists. First we load the table length (two words) into the accumulator. Next we load the starting address of the source table, using the LDA instruction. Then we load the data into the accumulator to be ANDed with the table. In the ANDMOV command, we specify the table destination, V3100.

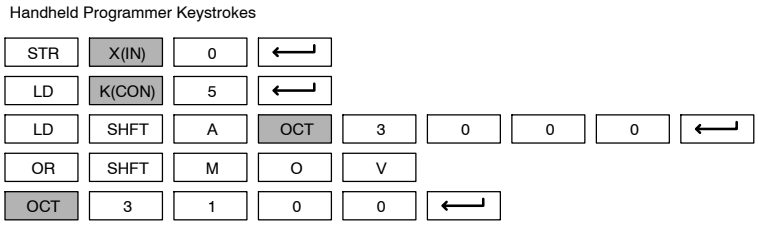




The example to the right shows a table of two words at V3000 and logically ORs it with K8888. The copy of the table at V3100 shows the result of the OR operation for each word.

The program to the right performs the ORMOV example above. It assumes that the data in the table at V3000 – V3001 already exists. First we load the table length (two words) into the accumulator. Next we load the starting address of the source table, using the LDA instruction. Then we load the data into the accumulator to be ORed with the table. In the ORMOV command, we specify the table destination, V3100.





The example to the right shows a table of two words at V3000 and logical XORs it with K3333. The copy of the table at V3100 shows the result of the XOR operation for each word.

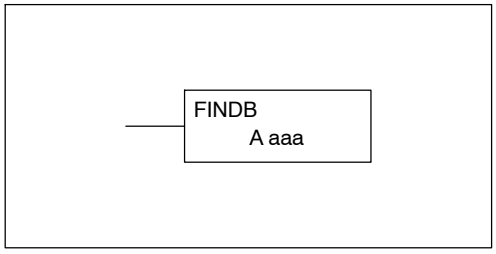
The ladder program example for the XORMOV is similar to the one above for the ORMOV. Just use the XORMOV instruction. On the handheld programmer, you must use the SHFT key and spell “XORMOV” explicitly.

Find Block (FINDB)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

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		DL450 Range
		aaa
V-memory	V	All (See p. 3-42)
V-memory	P	All (See p. 3-42)

Discrete Bit Flags	Description
SP53	on when the Find Block instruction was executed but did not find the block of data in table specified

The steps listed below are the steps necessary to program the Find Block function.

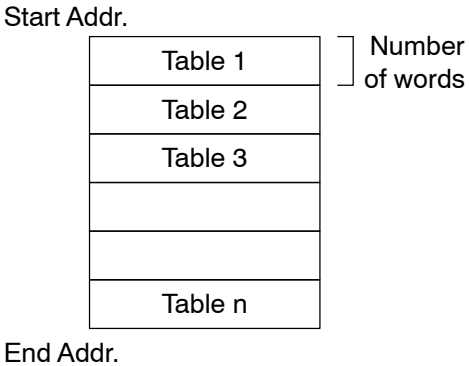
Step 1: — Load the number of bytes in the block to be located. This parameter must be a HEX value, 0 to FF.

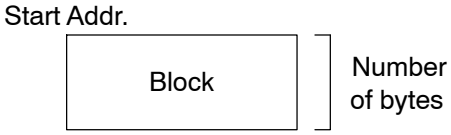
Step 2: — Load the length of a table (number of words) to be searched. The Find Block will search multiple tables that are adjacent in V-memory. This parameter must be a HEX value, 0 to FF.

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.



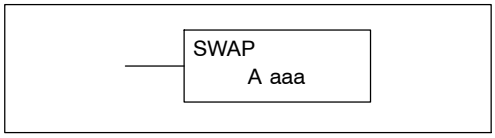


Swap (SWAP)

430	440	450
✗	✗	✓

DS	HPP
✓	✓

The Swap instruction exchanges the data in two tables of equal length.



The following description applies to both the Set Bit and Reset Bit table instructions.

Step 1: — Load the length of the tables (number of V-memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF. Remember that the tables must be of equal length.

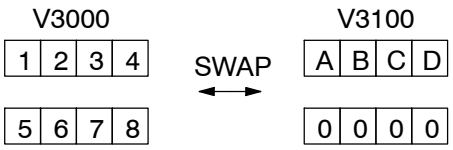
Step 2: — Load the starting V-memory location for the first table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

Step 3: —Insert the Swap instruction. This specifies the starting address of the second table. 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.

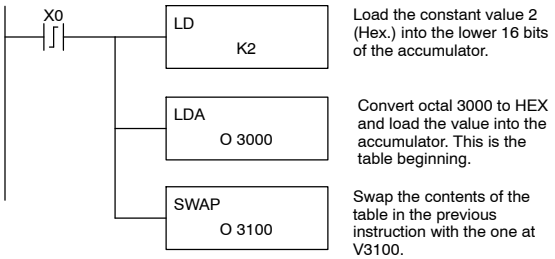
Operand Data Type		DL450 Range
		aaa
V-memory	V	All (See p. 3-42)

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.

DirectSOFT



Handheld Programmer Keystrokes

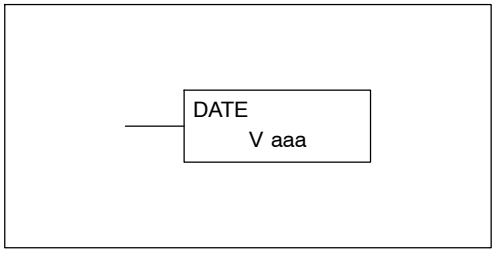
STR	X(IN)	SHFT	P	D	SHFT	0	↵			
LD	K(CON)	2	↵							
LD	SHFT	A	OCT	3	0	0	0	↵		
SHFT	S	W	A	P	OCT	3	1	0	0	↵

Clock/Calender Instructions

Date (DATE)

430	440	450	DS	HPP
✗	✓	✓	✓	✓

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).

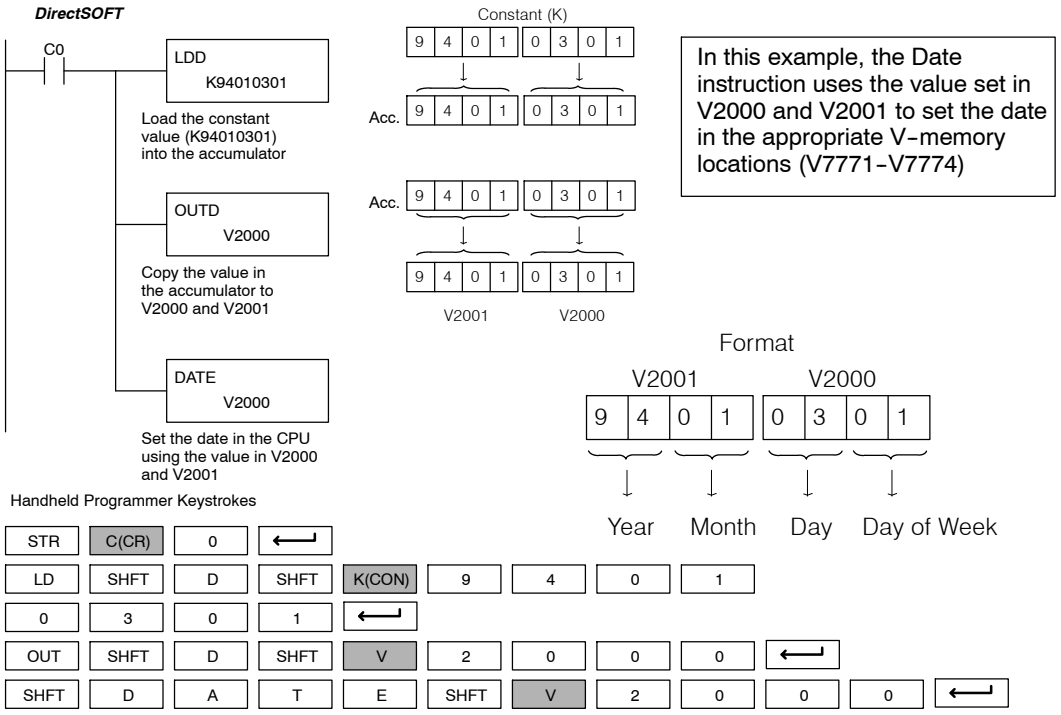


Date	Range	V Memory Location (BCD) (READ Only)
Year	0-99	V7774
Month	1-12	V7773
Day	1-31	V7772
Day of Week	0-06	V7771

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		DL440 Range	DL450 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-42)

In the following example, when C0 is on, the constant value (K94010301) is loaded into the accumulator using the Load Double instruction (C0 should be a contact from a one shot (PD) instruction). The value in the accumulator is output to V2000 using the Out Double instruction. The Date instruction uses the value in V2000 to set the date in the CPU.

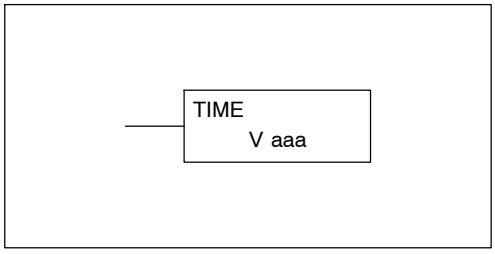


Time (TIME)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

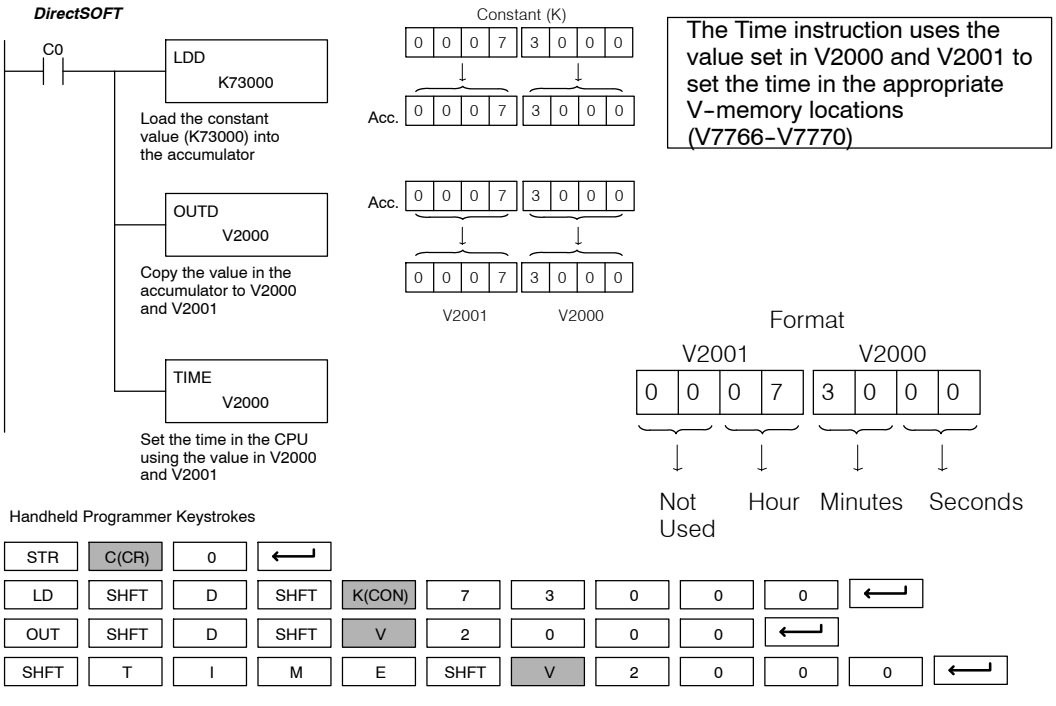
The Time instruction can be used to set the time (24 hour clock) in the CPU. The instruction requires two consecutive V-memory locations (Vaaa) which are used to *set the time*. If the values in the specified locations are not valid, the time will not be set. The current time can be read from V-memory 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		DL440 Range	DL440 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-41)

In the following example, when C0 is on, the constant value (K73000) is loaded into the accumulator using the Load Double instruction (C0 should be a contact from a one shot (PD) instruction). The value in the accumulator is output to V2000 using the Out Double instruction. The TIME instruction uses the value in V2000 to set the time in the CPU.



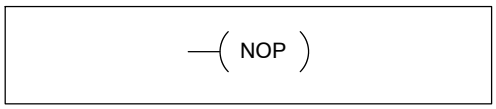
CPU Control Instructions

No Operation (NOP)

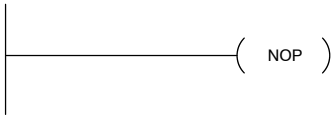
430	440	450
✗	✓	✓

DS	HPP
✓	✓

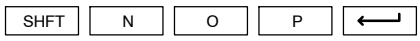
The No Operation is an empty (not programmed) memory location. These instructions are just place-holders in the program. So, you will not need to program them, because they automatically appear after the end of the program.



DirectSOFT



Handheld Programmer Keystrokes

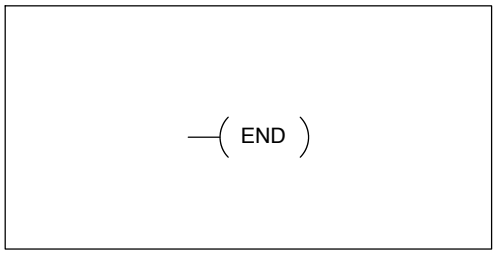


End (END)

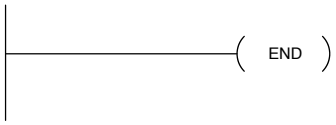
430	440	450
✓	✓	✓

DS	HPP
✓	✓

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.



DirectSOFT



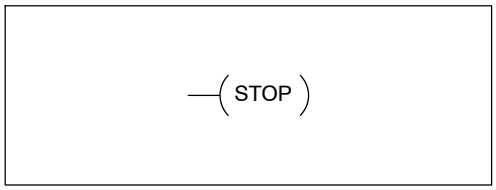
Handheld Programmer Keystrokes

Stop (STOP)

430	440	450
✓	✓	✓

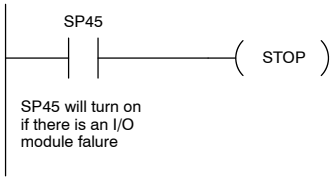
DS	HPP
✓	✓

The Stop instruction changes the operational mode of the CPU from Run to Program (Stop) mode. This instruction is typically used to stop PLC operation in a shutdown condition such as a I/O module failure.

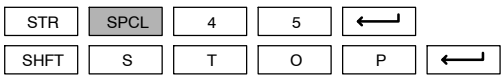


In the following example, when SP45 comes on indicating a I/O module failure, the CPU will stop operation and switch to the program mode.

DirectSOFT



Handheld Programmer Keystrokes

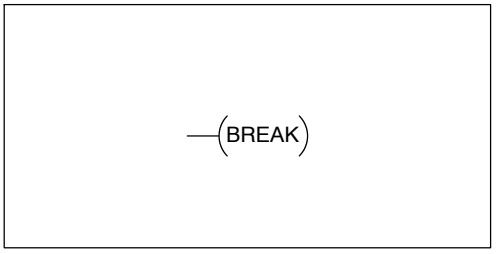


Break (BREAK)

430	440	450
✗	✓	✓

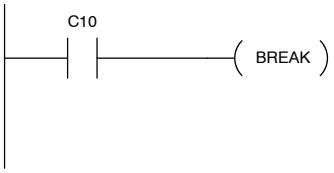
DS	HPP
✓	✓

The Break instruction changes the operational mode of the CPU from Run to the 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.



In the following example when C10 turns on, the CPU will stop operation and switch to the Test program mode.

DirectSOFT



Handheld Programmer Keystrokes

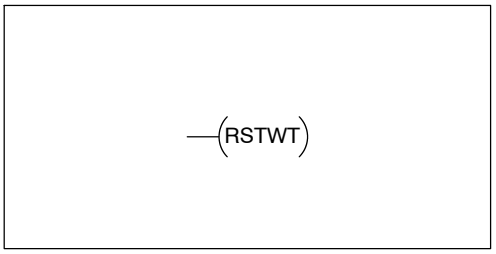
STR	C(CR)	1	0	↵		
SHFT	B	R	E	A	K	↵

Reset Watch Dog Timer (RSTWT)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

The Reset Watch Dog Timer instruction resets the CPU scan timer. The default setting for the watch dog timer is 200ms. Scan times very seldom exceed 200ms, but it is possible. For/next loops, subroutines, interrupt routines, and table instructions can be programmed such that the scan becomes longer than 200ms. When instructions are used in a manner that could exceed the watch dog timer setting, this instruction can be used to reset the timer.



A software timeout error (E003) will occur and the CPU will enter the program mode if the scan time exceeds the watch dog timer setting. Placement of the RSTWT instruction in the program is very important. The instruction has to be executed before the scan time exceeds the watch dog timer's setting.

If the scan time is consistently longer than the watch dog timer's setting, the timeout value may be permanently increased from the default value of 200ms by AUX 55 on the HPP or the appropriate auxiliary function in your programming package. This eliminates the need for the RSTWT instruction.

In the following example the CPU scan timer will be reset to 0 when the RSTWT instruction is executed. See the For/Next instruction for a detailed example.

DirectSOFT

Handheld Programmer Keystrokes

SHFT	R	S	T	W	T	↵

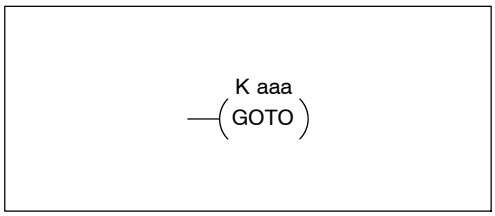
Program Control Instructions

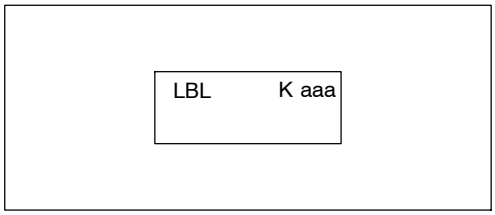
Goto/Label (GOTO/LBL)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

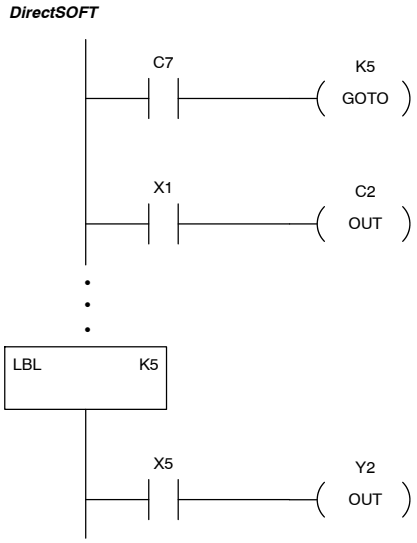
The Goto/Label skips all instructions between the Goto and the corresponding LBL instruction. The operand value for the Goto and the corresponding LBL instruction are the same. The logic between Goto and LBL instruction is not executed when the Goto instruction is enabled. Up to 128 Goto instructions and 64 LBL instructions can be used in the program.





Operand Data Type		DL440 Range	DL450 Range
		aaa	aaa
Constant	K	1-FFFF	1-FFFF

In the following example, when C7 is on, all the program logic between the Goto and the corresponding LBL instruction (designated with the same constant Kaaa value) will be skipped. The instructions being skipped will not be executed by the CPU.



Handheld Programmer Keystrokes

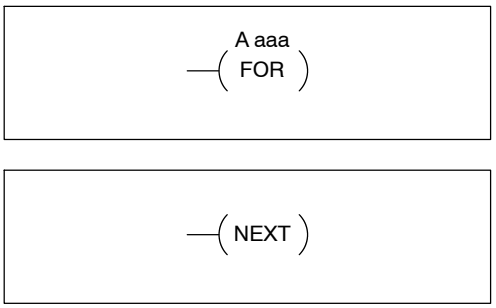
STR	C(CR)	7	↵					
SHFT	G	O	T	O	SHFT	K(CON)	5	↵
STR	X(IN)	1	↵					
OUT	C(CR)	2	↵					
•								
•								
SHFT	L	B	L	SHFT	K(CON)	5	↵	
STR	X(IN)	5	↵					
OUT	Y(OUT)	2	↵					

For/Next (FOR/NEXT)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

The For and Next instructions are used to execute a section of ladder logic between the For and Next instructions 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.

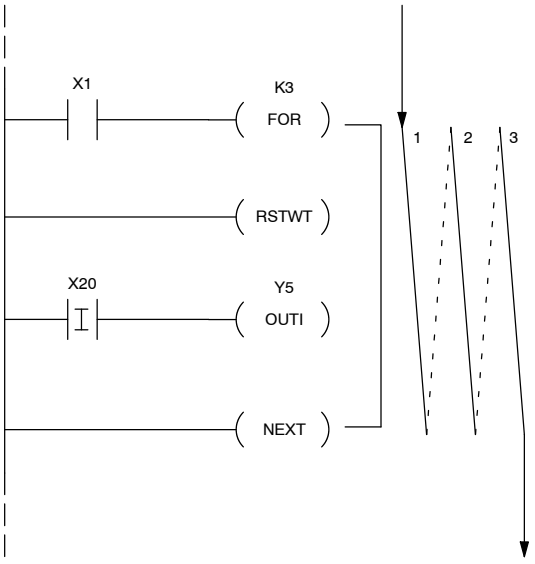


For / Next instructions cannot be nested. Up to 64 For / Next loops may be used in a program. If the maximum number of For / Next loops is exceeded, error E413 will occur. The normal I/O update and CPU housekeeping is suspended while executing the For / Next loop. The program scan can increase significantly, 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		DL440 Range	DL450 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-42)
Constant	K	1-9999	1-9999

In the following example, when X1 is on, the application program inside the For / Next loop will be executed three times. If X1 is off the program inside the loop will not be executed. The immediate instructions may or may not be necessary depending on your application. Also, The RSTWT instruction is not necessary if the For / Next loop does not extend the scan time larger the Watch Dog Timer setting. For more information on the Watch Dog Timer, refer to the RSTWT instruction.

DirectSOFT



Handheld Programmer Keystrokes

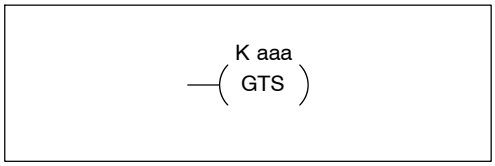
STR	X(IN)	1	↵				
SHFT	F	O	R	SHFT	K(CON)	3	↵
SHFT	R	S	T	W	T	↵	
STR	SHFT	I	SHFT	X(IN)	2	0	↵
Y(OUT)	SHFT	I	SHFT	Y(OUT)	5	↵	
SHFT	N	E	X	T	↵		

Goto Subroutine (GTS)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

The Goto Subroutine instruction allows a section of ladder logic to be placed outside the main body of the program execute only when needed. There can be a maximum of 192 (DL440) and an unlimited amount for DL450 GTS instructions.



Upon completion of executing the subroutine, program execution returns to the main program immediately after the GTS instruction. 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.

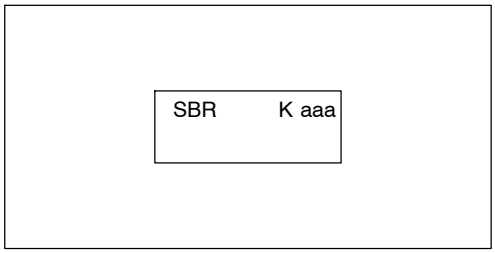
Operand Data Type		DL440 Range	DL450 Range
		aaa	aaa
Constant	K	1-FFFF	1-FFFF

Subroutine (SBR)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

The subroutine label and all associated logic is placed after the End statement in the program. There can be a maximum of 64 (DL440) and 256 (DL450) SBR instructions used in a 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.

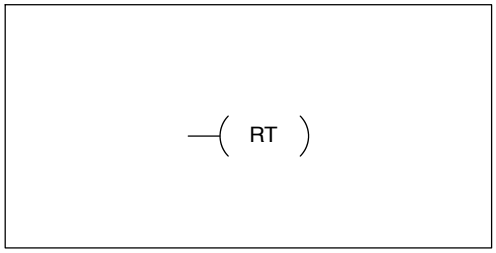
Operand Data Type		DL440 Range	DL450 Range
		aaa	aaa
Constant	K	1-FFFF	1-FFFF

Subroutine Return (RT)

430	440	450
✗	✓	✓

DS	HPP
✓	✓

When a Subroutine Return is executed in the subroutine the CPU will return to the point in the main body of the program from which it was called. The Subroutine Return is used as termination of the subroutine which must be the last instruction in the subroutine and is a stand alone instruction (no input contact on the rung).

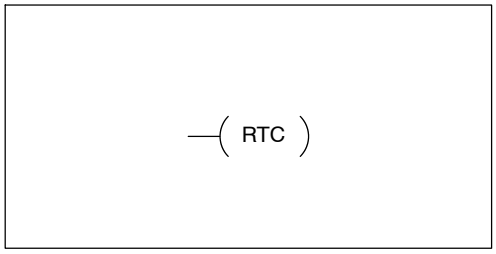


Subroutine Return Conditional (RTC)

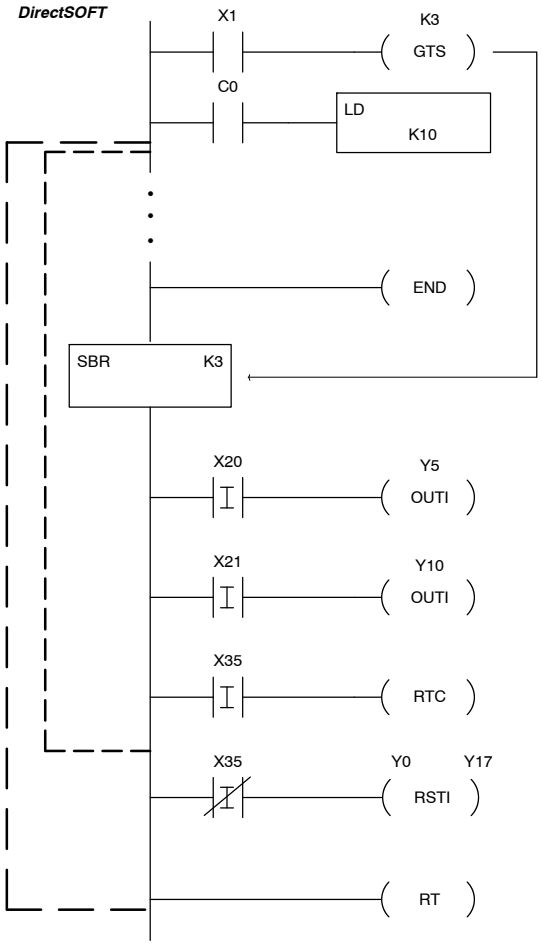
430	440	450
✗	✓	✓

DS	HPP
✓	✓

The Subroutine Return Conditional instruction is a optional instruction used with a input contact to implement a conditional return from the subroutine. The Subroutine Return (RT) is still required for termination of the Subroutine.



In the following example, when X1 is on, Subroutine K3 will be called. The CPU will jump to the Subroutine Label K3 and the ladder logic in the subroutine will be executed. If X35 is on the CPU will return to the main program at the RTC instruction. If X35 is not on Y0–Y17 will be reset to off and then the CPU will return to the main body of the program.



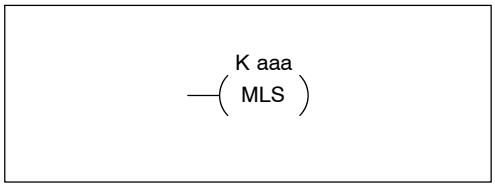
Handheld Programmer Keystrokes

STR	X(IN)	1	↵					
SHFT	G	T	S	SHFT	K(CON)	3	↵	
⋮								
END	↵							
SHFT	S	B	R	SHFT	K(CON)	3	↵	
STR	SHFT	I	SHFT	X(IN)	2	0	↵	
OUT	SHFT	I	SHFT	Y(OUT)	5	↵		
STR	SHFT	I	SHFT	X(IN)	2	1	↵	
OUT	SHFT	I	SHFT	Y(OUT)	1	0	↵	
STR	SHFT	I	SHFT	X(IN)	3	5	↵	
SHFT	R	T	C	↵				
STR	NOT	SHFT	I	SHFT	X(IN)	3	5	↵
RST	SHFT	I	SHFT	Y(OUT)	0	Y(OUT)	1	7	↵
SHFT	R	T	↵					

Master Line Set (MLS)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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 a 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.

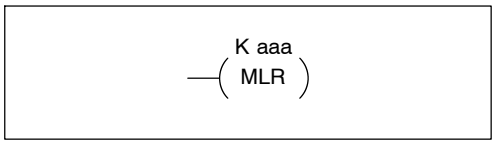


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Constant	K	1-7	1-7	1-7

Master Line Reset (MLR)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

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		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Constant	K	0-6	0-6	1-6

Understanding Master Control Relays

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

The Master Line Set (MLS) and Master Line Reset (MLR) instructions allow you to quickly control the power flow for 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.

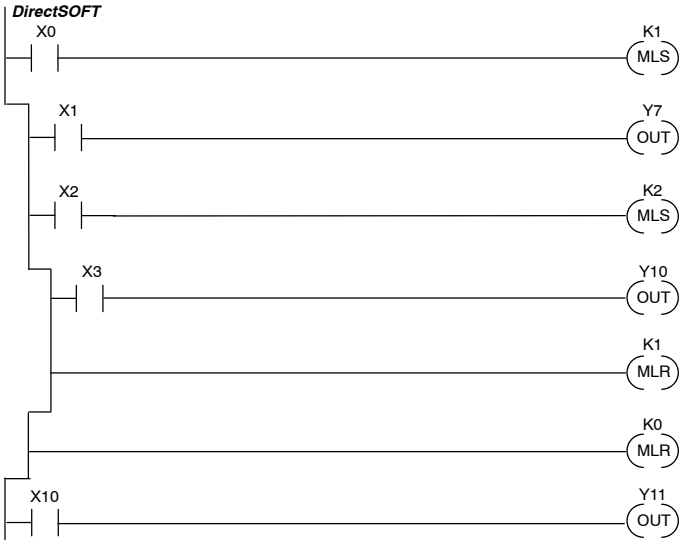
MLS/MLR Example

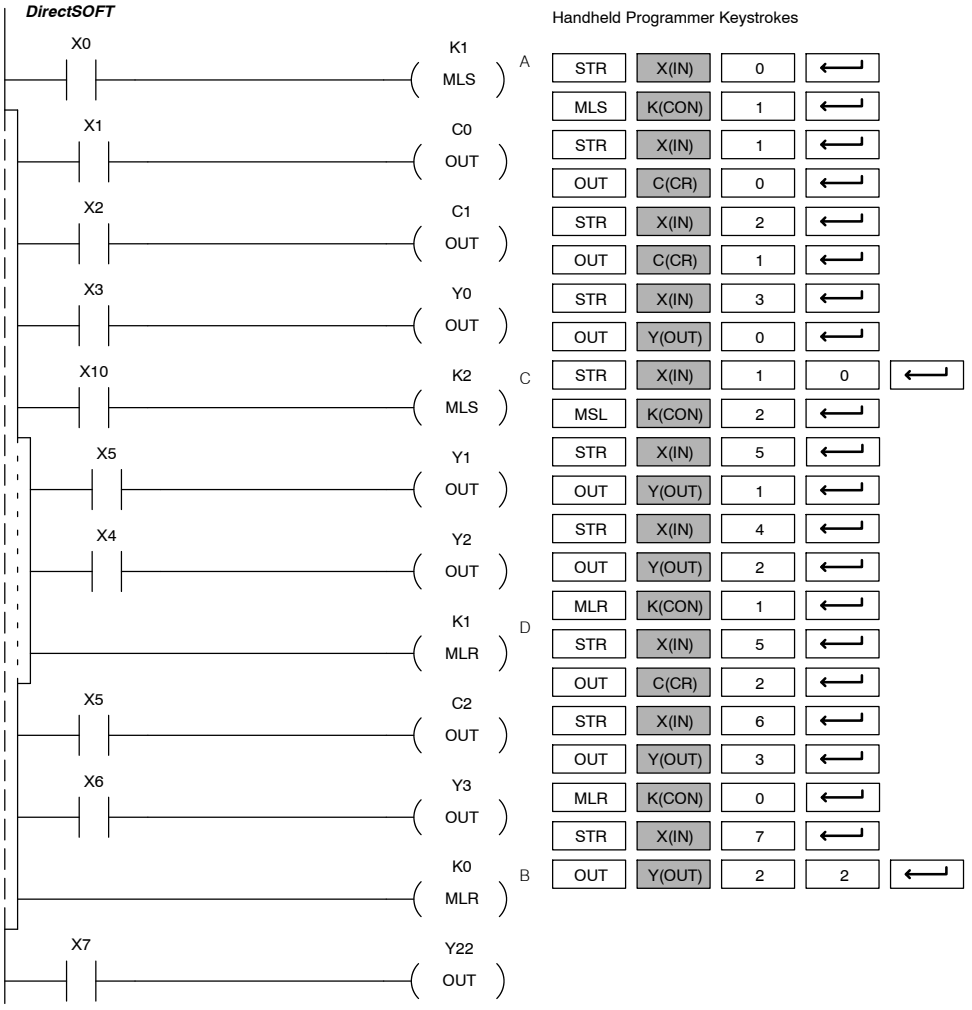
In the following MLS/MLR example, logic between the first MLS K1 (A) and MLR K0 (B) will only have power flow present at the power rail if input X0 is on. (Note, if X0 is off the logic will still be scanned, but there is no power flow.) The logic between the MLS K2 (C) and MLR K1 (D) will only have power flow present at the power rail if input X10 and X0 are on. The last rung is not controlled by either of the MLS coils and always has power flow present at the beginning of the rung.

Remember, the MLS / MLR instructions control the power flow between the power rails. It does not control the execution of the instructions. The instructions are still executed, but since there is no power flow, the logic cannot turn on the output coils. Consider the following case for our example.

1. X0 is off, which means that there is no power flow at the second rung.
2. You use a programming device to turn on C0.

Since there is no power flow at the second rung (STR X1, OUT C0), then you would expect that C0 would remain on, since you turned it on with the programming device. However, the MLS instruction does not mean that the instructions within the zone of control are not executed. They *are* in fact executed, but with no power flow. So in our case, C0 would be turned off when the rung was executed. This is because the CPU sees that there is no power flow present at the rung. When it executes this rung, it will turn off C0.


DirectSOFT
X0
K1
MLS
When contact X0 is on, logic under the first MLS will be executed.
X1
Y7
OUT
X2
K2
MLS
When contact X0 and X2 are ON, logic under the second MLS will be executed.
X3
Y10
OUT
K1
MLR
The MLR instructions note the end of the Master Control area.
K0
MLR
X10
Y11
OUT


DirectSOFT
X0
K1
MLS
A
X1
C0
OUT
X2
C1
OUT
X3
Y0
OUT
X10
K2
MLS
C
X5
Y1
OUT
X4
Y2
OUT
K1
MLR
D
X5
C2
OUT
X6
Y3
OUT
K0
MLR
B
X7
Y22
OUT
Handheld Programmer Keystrokes
STR X(IN) 0 ↵
MLS K(CON) 1 ↵
STR X(IN) 1 ↵
OUT C(CR) 0 ↵
STR X(IN) 2 ↵
OUT C(CR) 1 ↵
STR X(IN) 3 ↵
OUT Y(OUT) 0 ↵
STR X(IN) 1 0 ↵
MSL K(CON) 2 ↵
STR X(IN) 5 ↵
OUT Y(OUT) 1 ↵
STR X(IN) 4 ↵
OUT Y(OUT) 2 ↵
MLR K(CON) 1 ↵
STR X(IN) 5 ↵
OUT C(CR) 2 ↵
STR X(IN) 6 ↵
OUT Y(OUT) 3 ↵
MLR K(CON) 0 ↵
STR X(IN) 7 ↵
OUT Y(OUT) 2 2 ↵

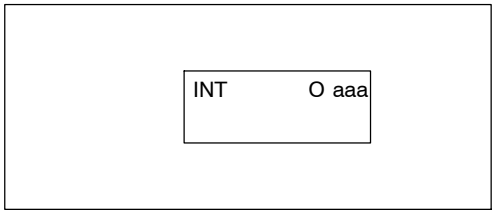
Interrupt Instructions

Interrupt (INT)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

The Interrupt instruction allows a section of ladder logic to be placed outside the main body of the program and executed when needed. Interrupts can be called from the program or an interrupt module can be installed in slot 0 to provide 8 interrupt inputs.



One interrupt module can be installed in a DL430 system (X0–X7) and two interrupt modules can be installed in a DL440 or DL450 System (X0–X7, and X20–X27). Remember, the interrupt modules consume 16 points.

The two software interrupts use interrupt #16 and #17 which means the hardware interrupts #16 and #17 and the software interrupt cannot be used together.
Typically, interrupts will be used in an application where a fast response to an input is needed or a program section needs to execute faster than the normal CPU scan. The interrupt label and all associated logic must be placed after the End statement in the program. When the interrupt routine is called from the interrupt module or software interrupt, the CPU will complete execution of the instruction it is currently processing in ladder logic, then execute the designated interrupt routine. There are two software interrupts and INT 16 and INT 17. Once the interrupt is serviced, the program execution will continue from where it was before the interrupt occurred.

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.

See the example program of a software interrupt.

Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Constant	O	0–7	0–17	0–17

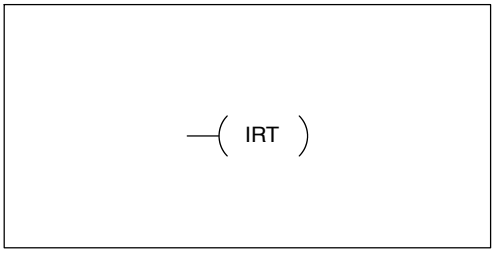
	Software		DL430		DL440/450	
	Interrupt Input	Interrupt Routine	Interrupt Input	Interrupt Routine	Interrupt Input	Interrupt Routine
	--	--	X0	INT 0	X0	INT 0
	--	--	X1	INT 1	X1	INT 1
	--	--	X2	INT 2	X2	INT 2
	--	--	X3	INT 3	X3	INT 3
	--	--	X4	INT 4	X4	INT 4
	--	--	X5	INT 5	X5	INT 5
	--	--	X6	INT 6	X6	INT 6
	--	--	X7	INT 7	X7	INT 7
	--	--	--	--	X20	INT 10
	--	--	--	--	X21	INT 11
	--	--	--	--	X22	INT 12
	--	--	--	--	X23	INT 13
2nd Module	--	--	--	--	X24	INT 14
	--	--	--	--	X25	INT 15
	V736 sets interrupt time	INT 16	--	--	X26(cannot be used along with s/w interrupt)	INT 16
	V737 sets interrupt time	INT 17	--	--	X27 (cannot be used along with s/w interrupt)	INT 17

Interrupt Return (IRT)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

When an Interrupt Return is executed in the interrupt routine the CPU will return to the point in the main body of the program from which it was called. The Interrupt Return is programmed as the last instruction in an interrupt routine and is a stand alone instruction (no input contact on the rung).

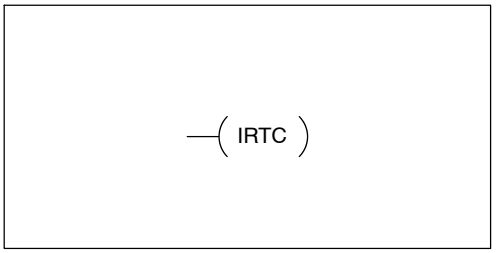


Interrupt Return Conditional (IRTC)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

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 still required to terminate the interrupt routine

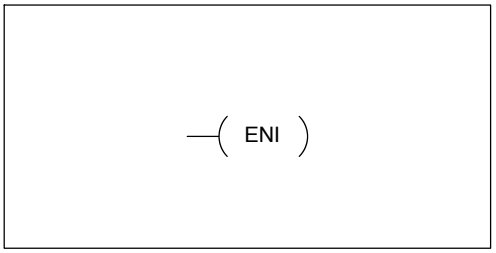


Enable Interrupts (ENI)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

The Enable Interrupt instruction is programmed in the main body of the application program (before the End instruction) to enable hardware or software interrupts. Once the coil has been energized interrupts will be enabled until the interrupt is disabled by the Disable Interrupt instruction.

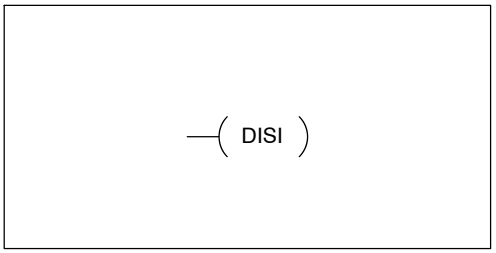


Disable Interrupts (DISI)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

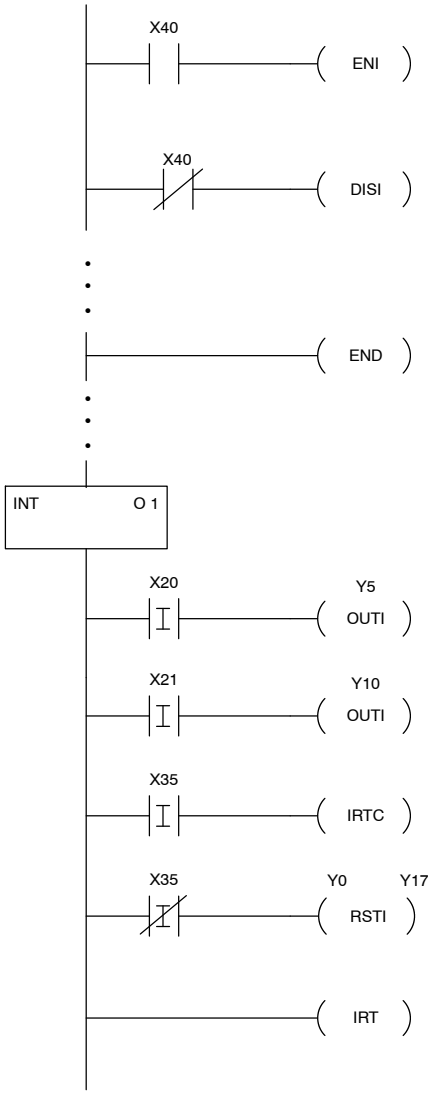
The Disable Interrupt instruction is programmed in the main body of the application program (before the End instruction) to disable both hardware or software interrupts. Once the coil has been energized interrupts will be disabled until the interrupt is enabled by the Enable Interrupt instruction.



Interrupt Example for Interrupt Module

In the following example, when X40 is on, the interrupts will be enabled. When X40 is off the interrupts will be disabled. When a interrupt signal X1 is received the CPU will jump to the interrupt label INT O 1. The application ladder logic in the interrupt routine will be performed. 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 main body of the program.

DirectSOFT



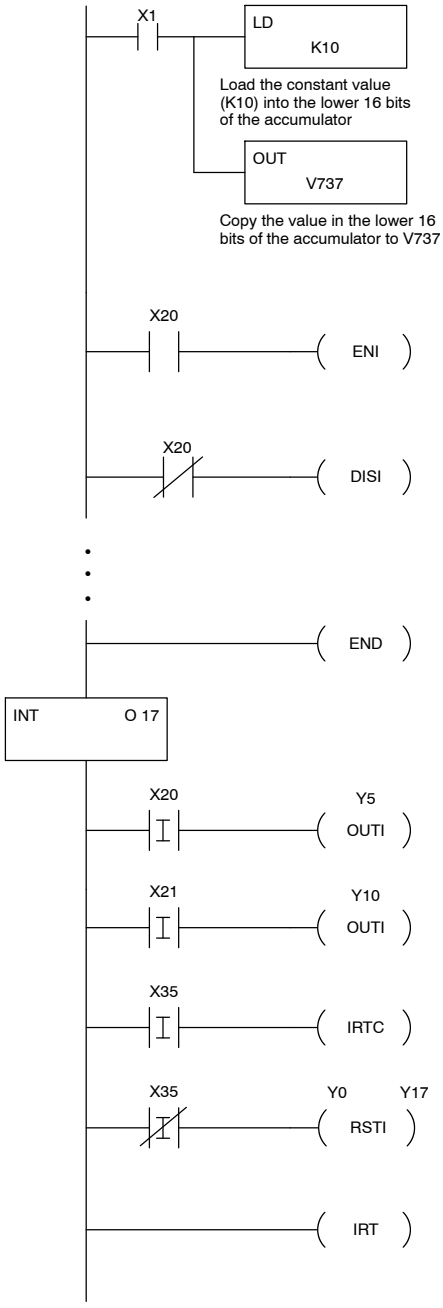
Handheld Programmer Keystrokes

STR	X(IN)	4	0	↵				
SHFT	E	N	I	↵				
STR	NOT	X(IN)	1	7	↵			
SHFT	D	I	S	I	↵			
•								
•								
END	↵							
SHFT	I	N	T	OCT	1	↵		
STR	SHFT	I	SHFT	X(IN)	2	0	↵	
OUT	SHFT	I	SHFT	Y(OUT)	5	↵		
STR	SHFT	I	SHFT	X(IN)	2	1	↵	
OUT	SHFT	I	SHFT	Y(OUT)	1	0	↵	
STR	SHFT	I	SHFT	X(IN)	3	5	↵	
SHFT	I	R	T	C	↵			
STR	NOT	SHFT	I	SHFT	X(IN)	3	5	↵
RST	SHFT	I	SHFT	Y(OUT)	0	Y(OUT)	1	7	↵
SHFT	I	R	T	↵				

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 O 17. 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 main body of 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.

DirectSOFT



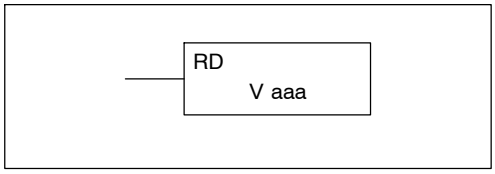
Handheld Programmer Keystrokes

STR	X(IN)	1	↵				
LD	K(CON)	1	0				
OUT	V	7	3	7	↵		
STR	X(IN)	2	0	↵			
SHFT	E	N	I	↵			
STR	NOT	X(IN)	2	0	↵		
SHFT	D	I	S	I	↵		
•							
•							
END	↵						
SHFT	I	N	T	OCT	1	7	↵
STR	SHFT	I	SHFT	X(IN)	2	0	↵
OUT	SHFT	I	SHFT	Y(OUT)	5	↵	
STR	SHFT	I	SHFT	X(IN)	2	1	↵
OUT	SHFT	I	SHFT	Y(OUT)	1	0	↵
STR	SHFT	I	SHFT	X(IN)	3	5	↵
SHFT	I	R	T	C	↵		
STR	NOT	SHFT	I	SHFT	X(IN)	3	5	↵
RST	SHFT	I	SHFT	Y(OUT)	0	Y(OUT)	1	7	↵
SHFT	I	R	T	↵				

Intelligent I/O Instructions

Read from Intelligent Module (RD)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP



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.

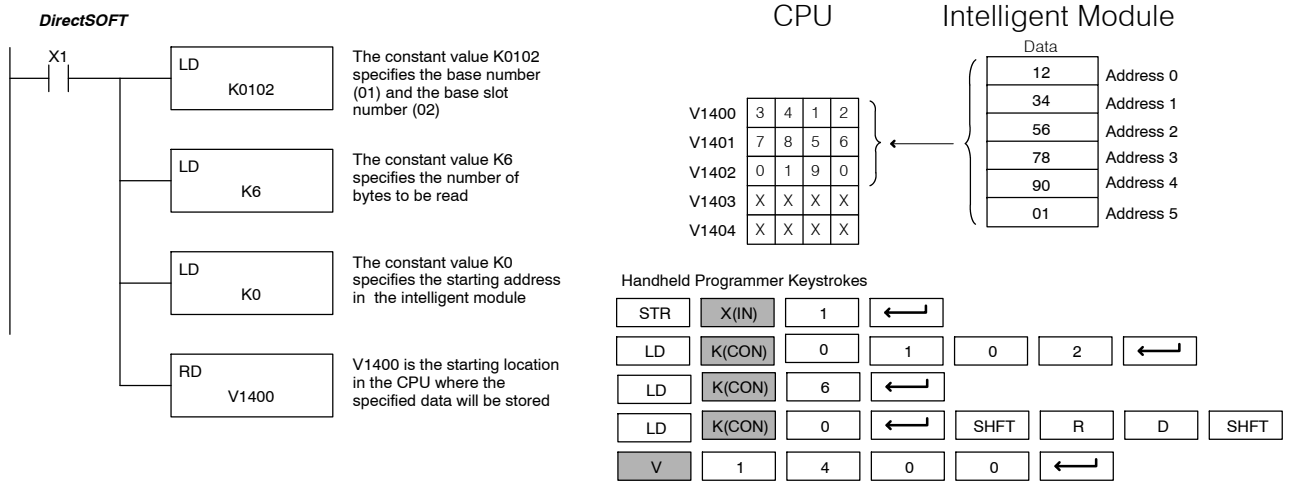
Helpful Hint: —Use the LDA instruction to convert an octal address to its HEX equivalent and load it into the accumulator when the hex format is required.

Operand Data Type	DL430 Range	DL440 Range	DL450 Range
	aaa	aaa	aaa
V-memory V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)

Discrete Bit Flags	Description
SP54	on when RX, WX, RD, WT instructions are executed with the wrong parameters.

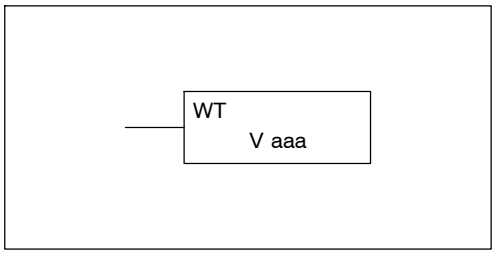
NOTE: Status flags are valid only until another instruction uses the same flag.

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.



Write to Intelligent Module (WT)

430	440	450	DS	HPP
✓	✓	✓	✓	✓



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 necessary 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 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.

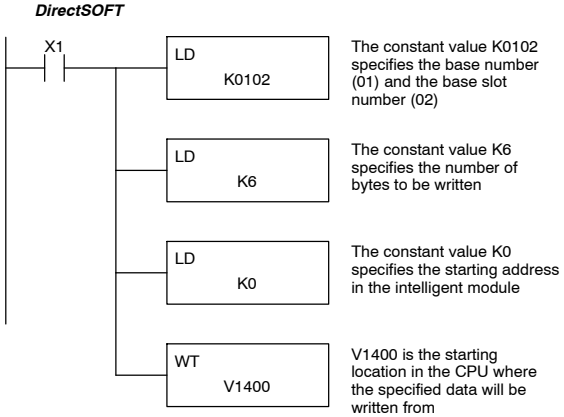
Helpful Hint: —Use the LDA instruction to convert an octal address to its HEX equivalent and load it into the accumulator when the hex format is required.

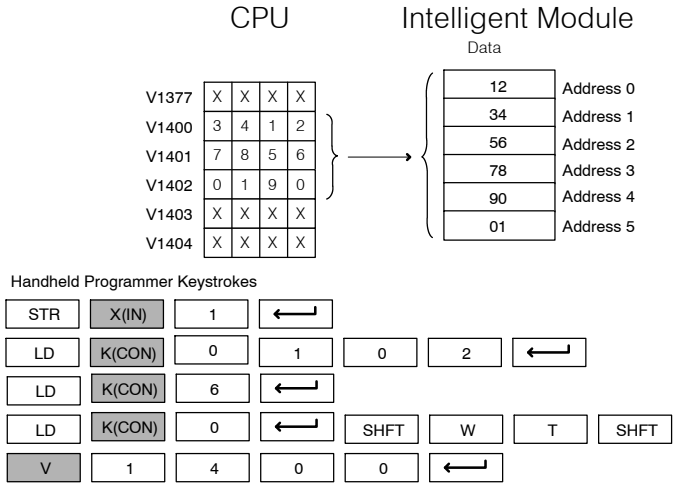
Operand Data Type	DL430 Range	DL440 Range	DL450 Range
	aaa	aaa	aaa
V-memory V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)

Discrete Bit Flags	Description
SP54	on when RX, WX, RD, WT instructions are executed with the wrong parameters.

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

In the following example, when X1 is on, the WT instruction will write six bytes of data to an intelligent module in base 1, slot 2 starting at address 0 in the intelligent module and copy the information from Vmemory locations V1400-V1402.



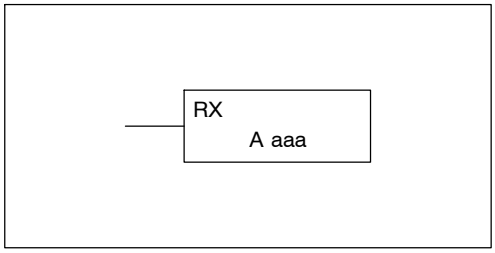


Network Instructions

Read from Network (RX)

✓ 430 ✓ 440 ✓ 450
✓ DS ✓ HPP

The Read from Network instruction is used by the master device (CPU with a DCM) on a network to read a block of data from another CPU. The function parameters are loaded into the first and second level of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Read from Intelligent module function.



Step 1: — Load the slave address (1-90 BCD) into the first byte and the slot number of the master DCM (0-7) into the second byte of the second level of the accumulator stack. (Note: address 0 is only valid for peer stations.)

Step 2: — Load the number of bytes (2-128 BCD, multiple of 2) to be transferred into the first level of the accumulator stack.

Step 3: — Load the address where you want to store the data in the master station. The address must be specified in HEX.

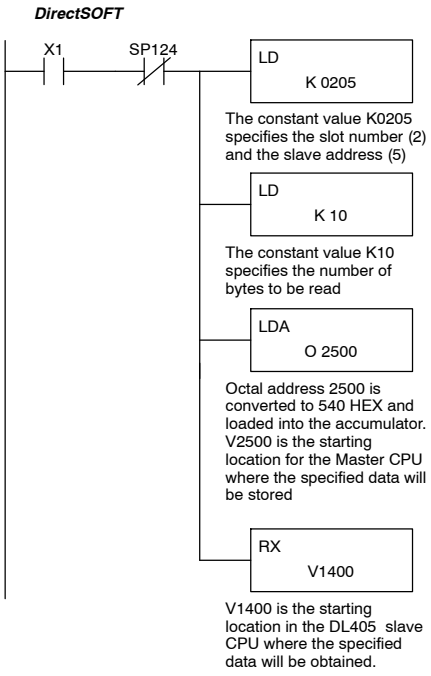
Step 4: — Insert the RX instruction and specify the starting V-memory location (Aaaa) in the slave CPU where the data will be obtained.

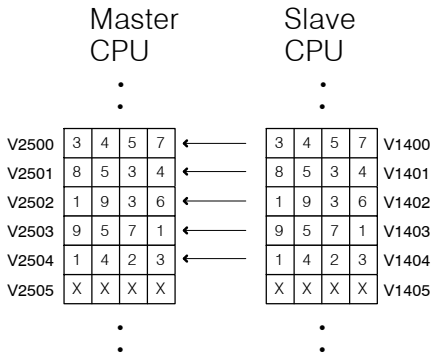
Helpful Hint: — For parameters that require HEX values, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Inputs	X	0-477	0-477	0-1777
Outputs	Y	0-477	0-477	0-1777
Control Relays	C	0-737	0-1777	0-3777
Stage	S	0-577	0-1777	0-1777
Timer	T	0-177	0-377	0-377
Counter	CT	0-177	0-177	0-377
Special Relay	SP	0-137, 320-617	0-137 320-717	0-137 320-717
Global I/O	GX	0-777	0-1777	0-2777
Program Memory	L$	0-3583	0-7679 (7.5K program mem.) 0-15871 (15.5K program mem.)	0-7679 (7.5K program mem.) 0-15871 (15.5K program mem.)
Scratchpad	Z	0-FFFF	0-FFFF	0-FFFF

NOTE: If you are using a CoProcessor, Share Data Network, or DCM module, please refer to their manuals for information on how to transfer data over the ***Direct***NET network.

In the following example, when X1 is on and the module busy relay SP124 (see special relays) is not on, the RX instruction will access a DCM operating as a master in slot 2. Ten consecutive bytes of data (V1400 – V1404) will be read from a CPU at station address 5 and copied into V-memory locations V2500–V2504 in the CPU with the master DCM.





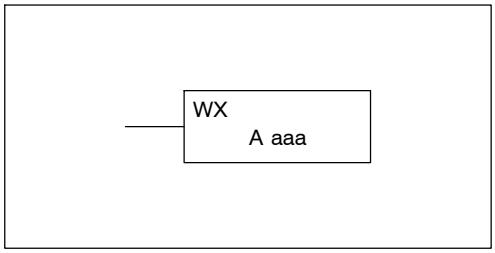
Handheld Programmer Keystrokes

STR	X(IN)	1	↵					
AND	NOT	SPCL	1	2	4	↵		
LD	K(CON)	0	2	0	5	↵		
LD	K(CON)	1	0	↵				
LD	SHFT	A	OCT	2	5	0	0	↵
SHFT	R	X	SHFT	V	1	4	0	0 ↵

Write to Network (WX)

✓ ✓ ✓
430 440 450
✓ ✓
DS HPP

The Write to Network instruction is used to write a block of data from the master device (CPU with a DCM) to 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 Write to Network function.



Step 1: — Load the slave address (1–90 BCD) into the first byte and the slot number of the master DCM (0–7) into the second byte of the second level of the accumulator stack. (Note: address 0 is only valid for peer stations.)

Step 2: — Load the number of bytes (2–128 BCD, multiple of 2) to be transferred into the first level of the accumulator stack.

Step 3: — Load the address where you want to obtain the data in the master station. The address must be specified in HEX.

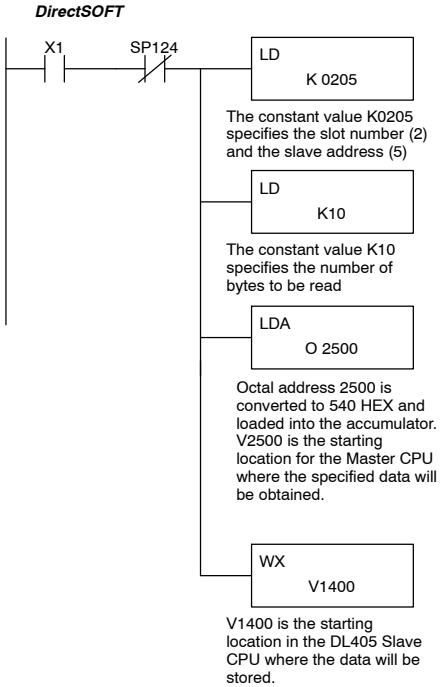
Step 4: — Insert the WX instruction and specify the starting V-memory location (Aaaa) in the slave CPU where the data will be stored.

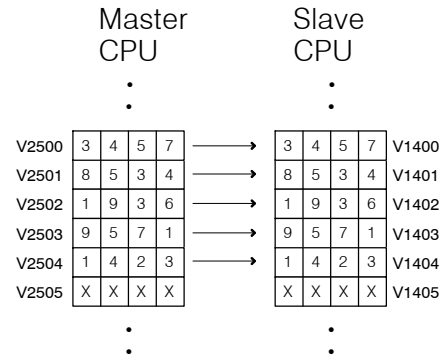
Helpful Hint: — For parameters that require HEX values, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type		DL430 Range	DL440 Range	DL450 Range
	A	**aaa**	**aaa**	**aaa**
V-memory	V	All (See p. 3-40)	All (See p. 3-41)	All (See p. 3-42)
Inputs	X	0-477	0-477	0-1777
Outputs	Y	0-477	0-477	0-1777
Control Relays	C	0-737	0-1777	0-3777
Stage	S	0-577	0-1777	0-1777
Timer	T	0-177	0-377	0-377
Counter	CT	0-177	0-177	0-377
Special Relay	SP	0-137, 320-617	0-137 320-717	0-137 320-717
Global I/O	GX	0-777	0-1777	0-2777
Program Memory	$	0-3585	0-7679 (7.5K program mem.) 0-15873 (15.5K program mem.)	0-7679 (7.5K program mem.) 0-15873 (15.5K program mem.)
Scratchpad	Z	0-FFFF	0-FFFF	0-FFFF

NOTE: If you are using a CoProcessor, Share Data Network, or DCM module, please refer to their manuals for information on how to transfer data over the DirectNET network.

In the following example when X1 is on and the module busy relay SP124 (see special relays) is not on, the WX instruction will access a DCM operating as a master in slot 2. 10 consecutive bytes of data is read from the CPU at station address 5 and copied to V-memory locations V1400-V1404 in the slave CPU.





Handheld Programmer Keystrokes

STR	X(IN)	1	↵					
AND	NOT	SPCL	1	2	4	↵		
LD	K(CON)	0	2	0	5	↵		
LD	K(CON)	1	0	↵				
LD	SHFT	A	OCT	2	5	0	0	↵
SHFT	W	X	SHFT	V	1	4	0	0 ↵

Message Instructions

System Errors and Fault Messages

The DL405 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.

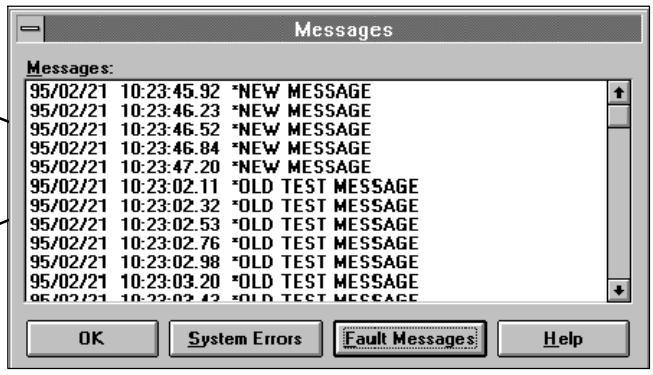
- **System Error Table** – both the DL430 and DL440 have several predefined system error codes. The DL430 can hold one error at a time. The DL440 can show up to 32 errors in an error table. When an error occurs, the error is loaded into the first available location. *Therefore, the most recent error 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 DL430 and DL440 also allow you to build your own error codes and messages. These are called Fault Messages. With the DL430, you can only build numeric error codes. With the DL440, 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 the Fault Message table as shown in ***Direct***SOFT. You cannot view the entire table at one time with the handheld programmer. Instead, the messages automatically appear on the handheld programmer display as they occur. The message will remain on the display as long as the Fault instruction is being executed. You can also use an Auxiliary function (5C) to view the messages one at a time. (More on the handheld programmer display later.)

DL440 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.



There are several instructions that can be used in combination to create these error codes and messages.

- FAULT – Fault
- DLBL – Data Label
- ACON – ASCII Constant
- NCON – Numeric Constant

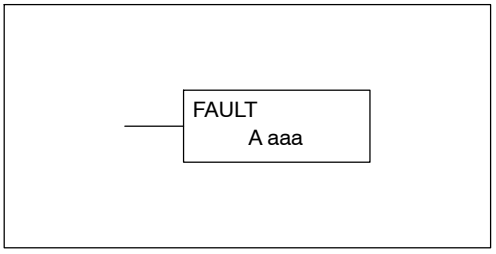
The next few pages provide details on these instructions. Also, at the end of this section, there are two examples that show how the instructions are used together.

Fault (FAULT)

430	440	450
✗	✓	✓

DS	HPP
✓	✗

In a DL440 or DL450, the Fault instruction is used to display a message or numeric error code on the handheld programmer, ***Direct***SOFT screen, or DV-1000 Operator Interface display. The message has a maximum of 23 characters and can be either V-memory data, numerical constant data or ASCII text.



To display a 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.

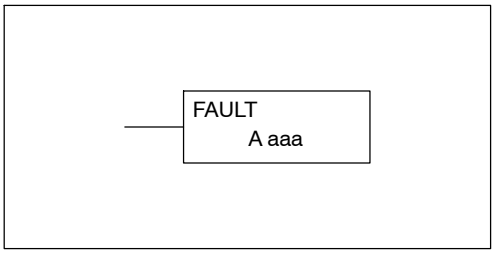
Operand Data Type		DL440 Range	DL450 Range
	A	**aaa**	**aaa**
V-memory	V	All (See p. 3-41)	All (See p. 3-42)
Constant	K	1-FFFF	1-FFFF

Fault (FAULT)

430	440	450
✓	✗	✗

DS	HPP
✓	✗

In a DL430, the Fault instruction is used to display a numeric error code on the handheld programmer, ***Direct***SOFT screen, or DV-1000 Operator Interface display.



The error code may be obtained from a V-memory location, or may be constructed by specifying a constant in the Fault instruction.

To display the value in a V-memory location, specify the V-memory location in the instruction. To display a numeric constant, specify the constant (K) value in the instruction.

Operand Data Type		DL430 Range
	A	**aaa**
V-memory	V	All (See p. 3-40)
Constant	K	1-FFFF

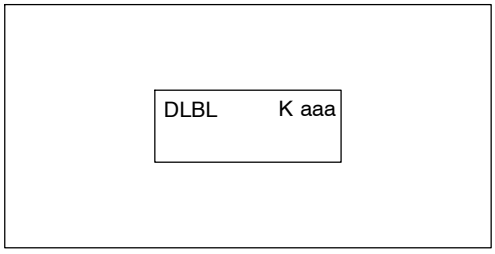
NOTE: The DL430 *does not* support the necessary instructions to build alphanumeric error messages. You can only build error codes by obtaining the code from a V-memory location, or by specifying a constant in the Fault instruction.

Data Label (DLBL)

430	440	450
✗	✓	✓

DS	HPP
✓	✗

The Data Label instruction marks the beginning of an ASCII/numeric data area and is typically used with ACON and NCON instructions. 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. Examples are shown later in this section.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Constant	K	--	1-FFFF	1-FFFF

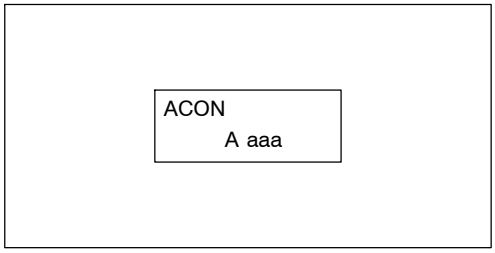
ASCII Constant (ACON)

430	440	450
✗	✓	✓

DS	HPP
✓	✗

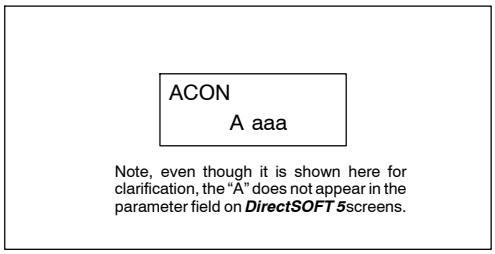
The ASCII Constant instruction is used with the DLBL instruction to store ASCII text for use with other instructions. The instruction is utilized differently between the handheld programmer and our ***Direct***SOFT programming software.

Handheld Programmer: With a handheld programmer, two ASCII characters can be stored in an ACON instruction. If only one character is stored in an ACON, a leading space will be printed in the Fault message.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
ASCII	A	--	0-9 A-Z	0-9 A-Z

DirectSOFT **Programming Software:**
If you're using ***Direct***SOFT, you can store up to 40 characters in an ACON. Also, you have a much wider range of characters that are supported. (See Appendix G for a complete listing of ASCII characters available.)



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
ASCII	A	--	(See ***DirectSOFT Manual***)	(See ***DirectSOFT Manual***)

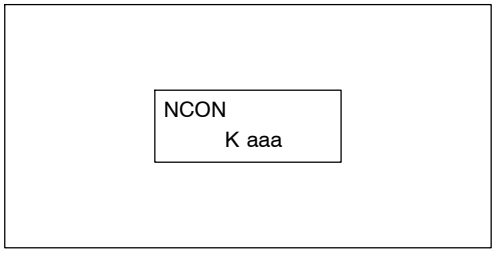
At first glance you may wonder why the instructions work differently in the two different programming tools. In reality, the instructions *do not* work differently. In ***Direct***SOFT, the 40-characters are actually broken down into multiple ACONs that contain 2 characters each when it is downloaded to the CPU. So, if you create the program with the software, and you examine the program mnemonics with a handheld (or even with ***Direct***SOFT), you would see multiple ACONs that contain 2 characters each. Examples are shown on the following pages.

Numeric Constant (NCON)

430	440	450
✗	✓	✓

DS	HPP
✓	✗

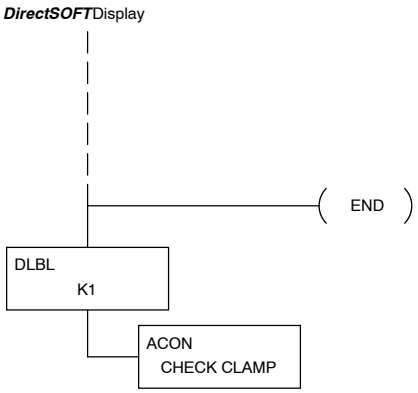
The Numeric Constant instruction is used with the DLBL instruction to store the HEX ASCII equivalent of numeric data for use with other instructions. Two digits can be stored in an NCON instruction.



Operand Data Type		DL440 Range	DL450 Range
		aaa	aaa
Constant	K	0-FFFF	0-FFFF

Using the Instructions to Build the Messages

The Fault instruction actually copies the messages into the appropriate Error table. However, it is important to understand how the DLBL, ACON, and NCON instructions are combined to *build* the message. The DLBL is placed after the END instruction, which signifies the end of the main program. The ACON and NCON instructions are placed within the DLBL area. The diagram shows an example.

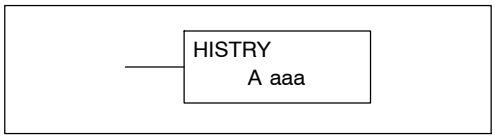


History (HISTRY)

430	440	450
✗	✗	✓

DS	HPP
✓	✗

The History instruction stores event history information in memory of the PLC.

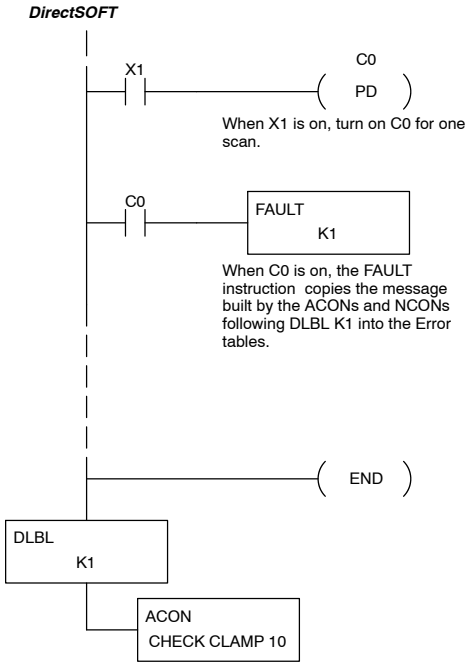


Important Information about FAULT Execution

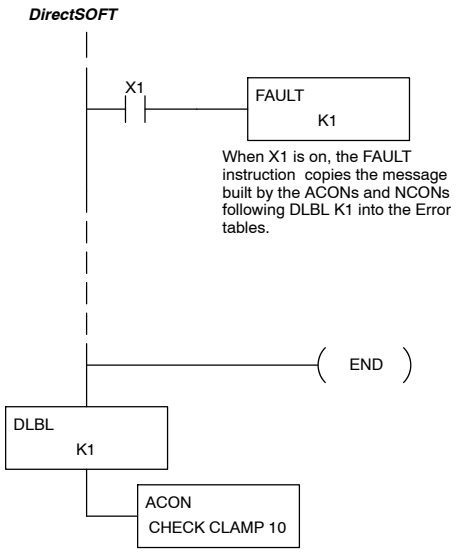
It is important to understand how the Error Message and Error Code tables work when the Fault instruction is executed. Each time the instruction is executed, the code or message is inserted into the table. Since the CPU scan is extremely fast, then it is easily possible to completely fill up a table with a single message. 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.



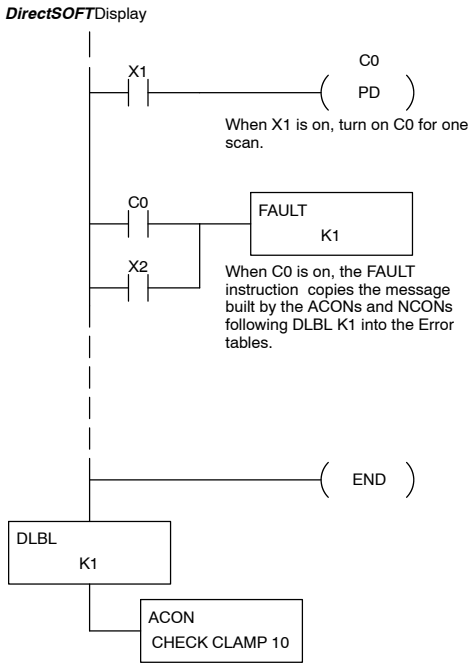
NOTE: This method *does not* work correctly with a handheld programmer. The message will never appear on the display. You *must* use the input contact (X1 in this example) to trigger the Fault instruction directly. The diagram shown here provides an example.



Obviously, you could combine other instructions with the PD instruction usage to develop logic that would work for both the handheld programmer *and* ***Direct***SOFT. For example, if it's possible in your application, you can have *two* input contacts that can trigger the FAULT instruction.

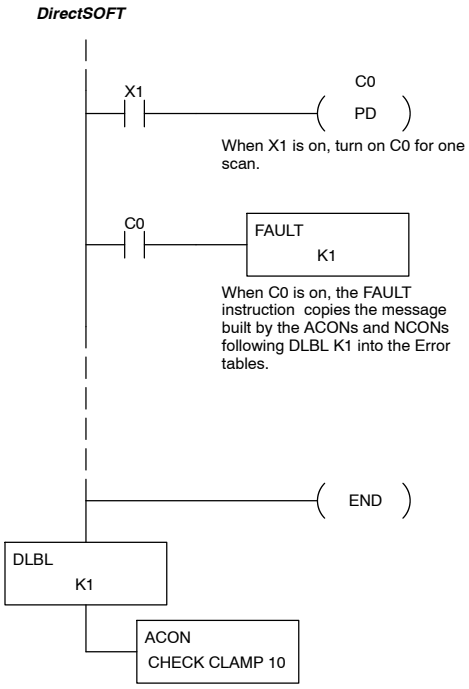
Consider the example shown to the right. If X1 comes on, then the message will only be copied into the table one time. You could see the message with ***Direct***SOFT, but you would not automatically see it on the handheld programmer display.

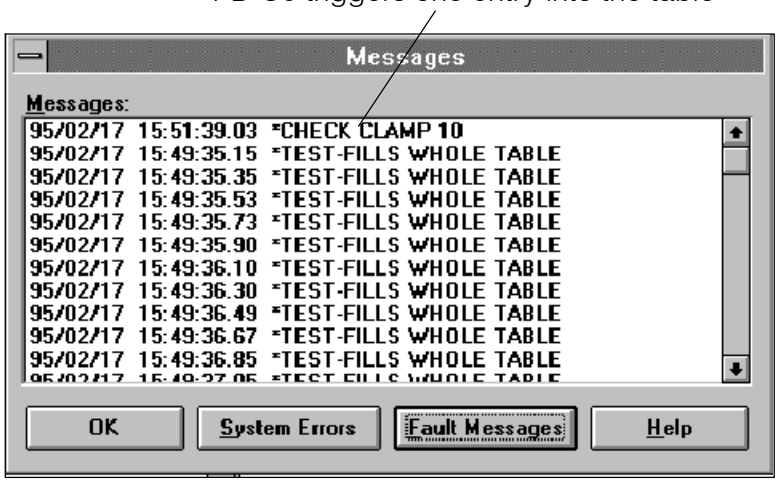
If X2 comes on, then the message would probably fill the entire table. Also, it would automatically appear on the handheld programmer.



*Direct*SOFT Example

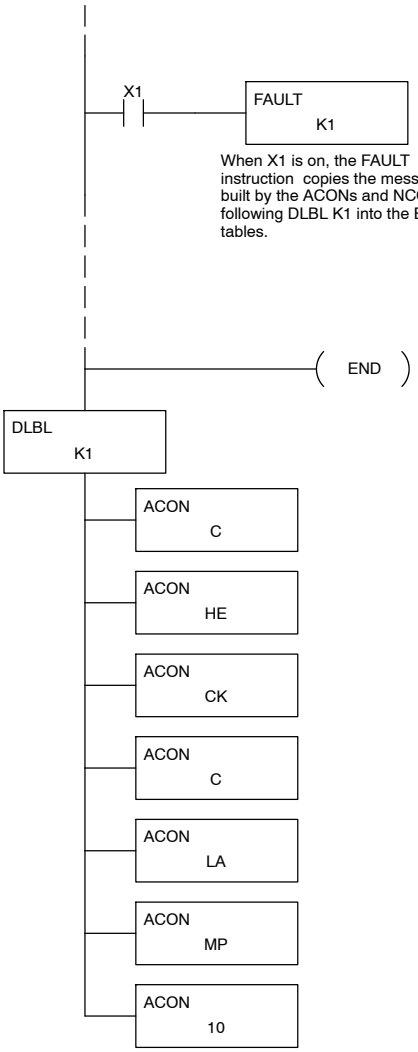
Since you can enter more characters in an ACON instruction in ***Direct***SOFT, it is very easy to build the entire message in one ACON instruction.



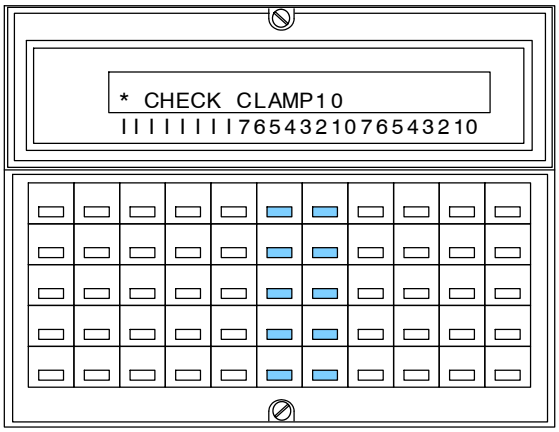


Handheld Programmer Example

Since you can only enter two characters per ACON with the handheld programmer, the program appears to be longer to get the same results. You'll also notice that we've organized the ACON contents slightly differently. For example, we've only included the character "C" in the first ACON. We have to do it this way in order to get the message to appear correctly. (Remember, a single character entered in an ACON with a handheld will have a blank space preceding it. So if your messages do not contain an even number of characters, you may have to play around with them to get the spacing just right.) Also, we have to use X1 to directly trigger the FAULT instruction. Otherwise, the message will never automatically appear on the handheld programmer display.



Message appears on handheld display



Asterisk (*) is automatically provided by the handheld operating system. It does not have to be entered with an ACON.

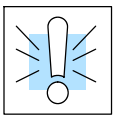
Handheld Programmer Keystrokes

STR	X(IN)	1	↵						
SHFT	F	A	U	L	T	SHFT	K(CON)	1	↵
•									
•									
END	↵								
SHFT	D	L	B	L	SHFT	K(CON)	1	↵	
SHFT	A	C	O	N	ASC	SHFT	C	↵	
SHFT	A	C	O	N	ASC	SHFT	H	E	↵
SHFT	A	C	O	N	ASC	SHFT	C	K	↵
SHFT	A	C	O	N	ASC	SHFT	C	↵	
SHFT	A	C	O	N	ASC	SHFT	L	A	↵
SHFT	A	C	O	N	ASC	SHFT	M	P	↵
SHFT	A	C	O	N	ASC	1	0	↵	

Clearing the Messages

You use different methods to clear the System Errors vs. the Fault Messages.

- System Errors — initialize the CPU's scratchpad memory
- Fault Message Table — clear the CPU program memory



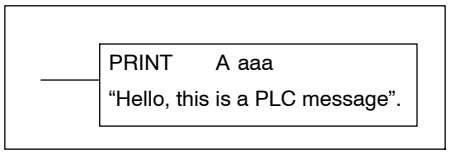
WARNING: If you initialize the scratchpad, you will also remove any retentive memory ranges that you may have changed.

Print Message (PRINT)

430	440	450
✓	✓	✓

DS	HPP
✓	✓

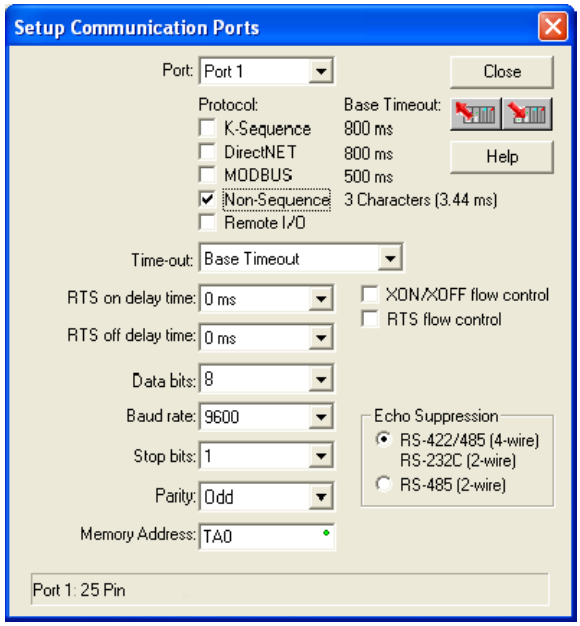
The Print Message instruction prints the embedded text or text/data variable message to the specified communications port (1, 2, or 3 on the DL450 CPU), which must have the communications port configured.



Data Type		DL450 Range
	A	aaa
Constant	K	1, 2, or 3

You may recall from the CPU specifications in Chapter 3 that the DL450's ports are capable of several protocols. To configure a port using the Handheld Programmer, use AUX 56 and follow the prompts, making the same choices as indicated below on this page. To configure a port in ***Direct***SOFT, choose PLC on the menu bar, then **Setup > Setup Secondary Comm Port**, the dialog below will appear.

- **Port:** From the port number list box at the top, choose "Port 1".
- **Protocol:** Click the check box to the left of "Non-sequence".



- **RTS Flow Control:** Choose the appropriate Request-to-Send option for your printer.
- **Data bits:** Choose the data bits
- **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 ***Direct***SOFT *to* use to store the port setup information. You will need to reserve 66 contiguous words in V-memory for this purpose. Select "Use for printing only".

Click the button indicated to send the Port 1 configuration to the CPU, and click **Close**. See Chapter 3 for port wiring information, in order to connect your printer to the DL450.

Ports 1 and 2 on the DL450 has standard RS232 levels, and should work with most printer serial input connections. Port 3 has RS422 levels, and will not work with most printers. You could use port 3 for long cables, converting the signals to RS232 at the other end to drive the printer.

Text element – this is used for printing character strings. The character strings are defined as the character (more than 0) ranged by the double quotation marks. Two hex numbers preceded by the dollar sign means an 8-bit ASCII character code. Also, two characters preceded by the dollar sign is interpreted according to the following table:

#	Character code	Description
1	$$	Dollar sign ($)
2	$”	Double quotation (”)
3	$L or $l	Line feed (LF)
4	$N or $n	Carriage return line feed (CRLF)
5	$P or $p	Form feed
6	$R or $r	Carriage return (CR)
7	$T or $t	Tab

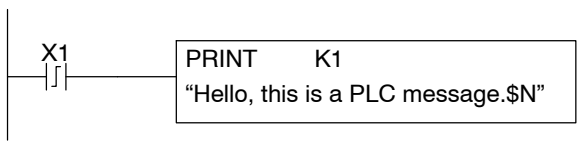
The following examples show various syntax conventions and the length of the output to the printer.

Example:

” ”	Length 0 without character
”A”	Length 1 with character A
” ”	Length 1 with blank
” $” ”	Length 1 with double quotation mark
” $ R $ L ”	Length 2 with one CR and one LF
” $ 0 D $ 0 A ”	Length 2 with one CR and one LF
” $ $ ”	Length 1 with one $ mark

In printing an ordinary line of text, you will need to include double quotation marks before and after the text string. Note that ***Direct***SOFT does not give error indications for syntax errors in the PRINT instruction. Therefore, it is important to test your PRINT instruction data during the application development.

The following example prints the message to port 1. We use a PD contact, which causes the message instruction to be active for just one scan. Note the $N at the end of the message, which produces a carriage return / line feed on the printer. This prepares the printer to print the next line, starting from the left margin.



Print the message to Port 1 when X1 makes an off-to-on transition.

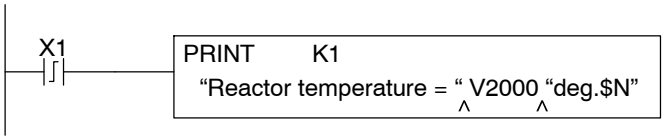
V-memory element – this is used for printing V-memory contents in the integer format or real format. Use V-memory number or V-memory number with ":" and data type. The data types are shown in the table below. The Character code must be in capital letters.

#	Character code	Description
1	none	16-bit binary (decimal number)
2	: B	4 digit BCD
3	: D	32-bit binary (decimal number)
4	: D B	8 digit BCD
5	: R	Floating point number (real number)
6	: E	Floating point number (real number with exponent)

Example:

V2000	Print binary data in V2000 for decimal number
V2000 : B	Print BCD data in V2000
V2000 : D	Print binary number in V2000 and V2001 for decimal number
V2000 : D B	Print BCD data in V2000 and V2001
V2000 : R	Print floating point number in V2000/V2001 as real number
V2000 : E	Print floating point number in V2000/V2001 as real number with exponent

Example: The following example prints a message containing text and a variable. The "reactor temperature" labels the data, which is at V2000. You can use the : B qualifier after the V2000 if the data is in BCD format, for example. The final string adds the units of degrees to the line of text, and the $N adds a carriage return / line feed. You must include spaces between the text string and the variable.



Print the message to Port 1 when X1 makes an off-to-on transition.

^ represents a space

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.

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 : ON:OFF	Prints the status of C00 in ON/OFF format
V2000.15 : BOOL	Prints the status of bit 15 in V2000 in TRUE/FALSE format

The maximum numbers of characters you can print is 128. The number of characters for each element is listed in the table below:

Element type	Maximum Characters
Text, 1 character	1
16 bit binary	6
32 bit binary	11
4 digit BCD	4
8 digit BCD	8
Floating point (real number)	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

The handheld programmer's mnemonic is "PRINT", followed by the DEF field.

Special relay flags SP112 through SP117 indicate the status of the DL450 CPU ports (busy, or communications error). See the appendix on special relays for a description.

NOTE: You must use the appropriate special relay in conjunction with the PRINT command to ensure the ladder program does not try to PRINT to a port that is still busy from a previous PRINT or WX or RX instruction.

6

Drum Instruction Programming

(DL450 CPU only)

In This Chapter. . . .

— Introduction
— Step Transitions
— Overview of Drum Operation
— Drum Control Techniques
— Drum Instructions

Introduction

Purpose

X	X	✓
430	440	450

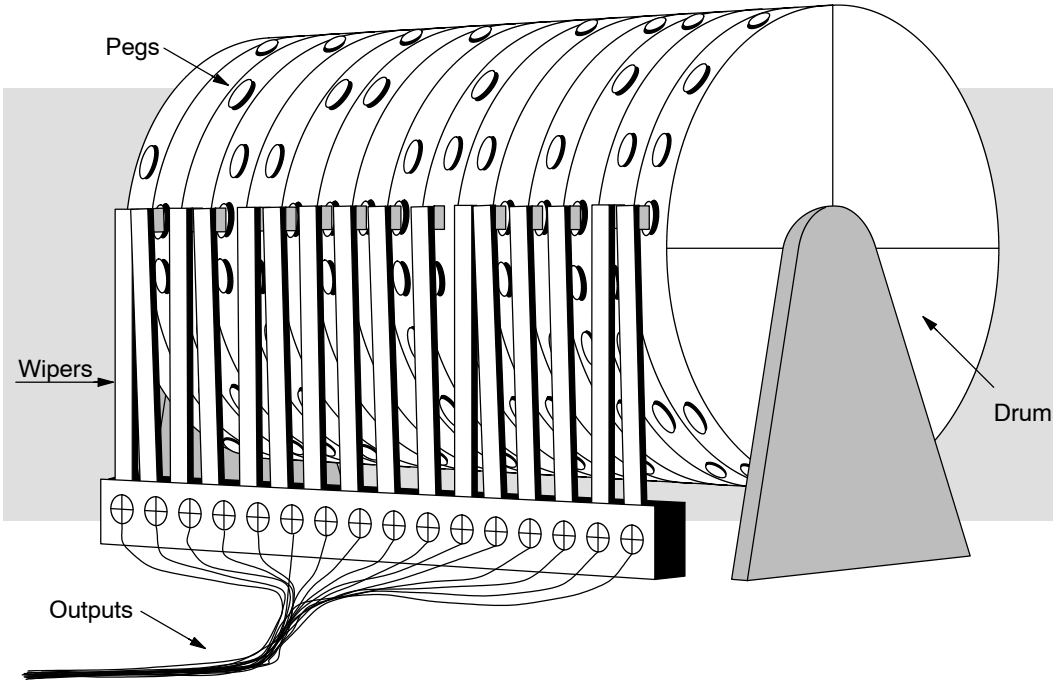
The four drum instructions available in the DL450 CPU electronically simulate an electro-mechanical drum sequencer. The instructions offer slight variations on the basic principle, which we describe first.

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 drum instructions by describing the original electro-mechanical drum pictured below. The mechanical **drum** generally has pegs on its curved surface. The pegs are populated in a particular **pattern**, representing a set of desired actions for machine control. A motor or solenoid rotates the drum a precise amount at specific times. During rotation, stationary wipers sense the presence of pegs (present = on, absent = off). This interaction makes or breaks electrical contact with the wipers, creating electrical **outputs** from the drum. The outputs are wired to devices on a machine for On/Off control.

Drums usually have a finite number of positions within one rotation, called **steps**. Each step represents some process step. At powerup, the drum **resets** to a particular step. The drum rotates from one step to the next based on a **timer**, or on some external **event**. During special conditions, a machine operator can manually increment the drum step using a **jog** control on the drum's drive mechanism. The contact closure of each wiper generates a unique on/off pattern called a **sequence**, designed for controlling a specific machine. Because the drum is circular, it automatically repeats the sequence once per rotation. Applications vary greatly, and a particular drum may rotate once per second, or as slowly as once per week.



Electronic drums provide the benefits of mechanical drums and more. For example, they have a **preset** feature that is impossible for mechanical drums: The preset function lets you move from the present step *directly* to any other step on command!

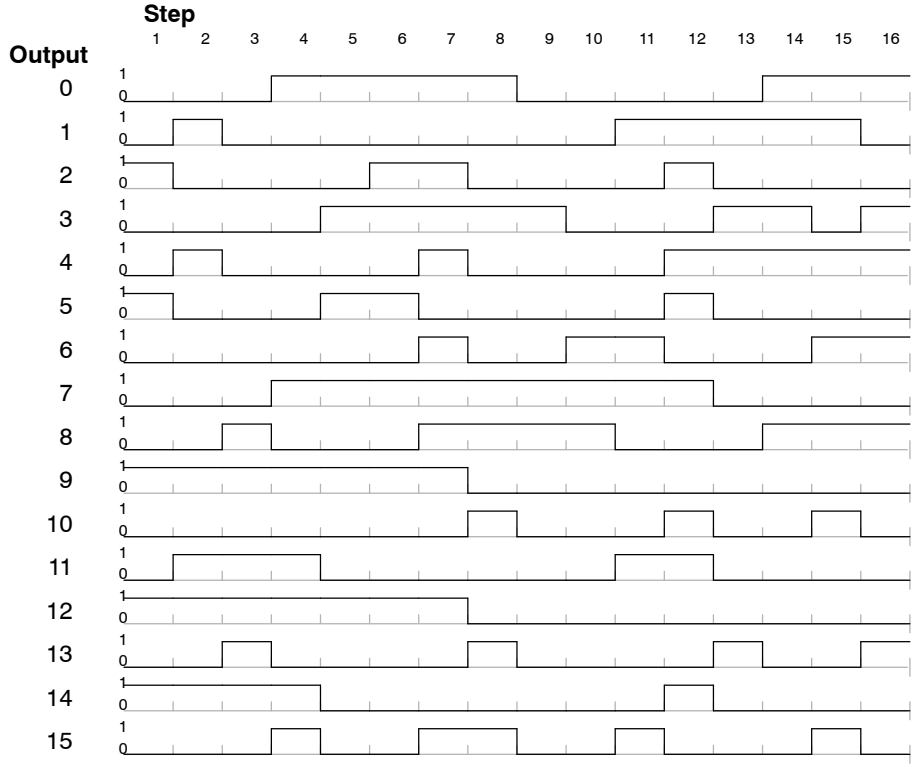
Drum Chart Representation

For editing purposes, the electronic drum is presented in chart form in ***Direct***SOFT and in this manual. Imagine slicing the surface of a hollow drum cylinder between two rows of pegs, then pressing it flat. Now you can view the drum as a chart as shown below. Each row represents a step, numbered 1 through 16. Each column represents an output, numbered 0 through 15 (to match word bit numbering). The solid circles in the chart represent pegs (On state) in the mechanical drum, and the open circles are empty peg sites (Off state).

	OUTPUTS															
STEP	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	○	●	○	●	○	○	●	○	○	○	●	○	○	●	○	○
2	○	●	○	●	●	○	●	○	○	○	○	●	○	○	●	○
3	○	●	●	●	●	○	●	●	○	○	○	○	○	○	○	○
4	●	●	○	●	●	○	●	○	●	○	○	○	○	○	○	●
5	○	○	○	●	○	○	●	○	●	○	●	○	●	○	○	●
6	○	○	○	●	○	○	●	○	●	○	●	○	●	●	○	●
7	●	○	○	●	○	○	●	●	●	●	○	●	●	●	○	●
8	●	○	●	○	○	●	○	●	●	○	○	○	●	○	○	●
9	○	○	○	○	○	○	○	●	●	○	○	○	●	○	○	○
10	○	○	○	○	○	○	○	●	●	●	○	○	○	○	○	○
11	●	○	○	○	●	○	○	○	○	●	○	○	○	○	●	○
12	○	●	○	○	●	●	○	○	●	○	●	●	○	●	●	○
13	○	○	●	○	○	○	○	○	○	○	○	●	●	○	●	○
14	○	○	○	○	○	○	○	●	○	○	○	●	●	○	●	●
15	●	○	○	○	○	●	○	●	○	●	○	●	○	○	●	●
16	○	○	●	○	○	○	○	●	○	●	○	●	●	○	○	●

Output Sequences

The mechanical drum sequencer derives its name from sequences of control changes on its electrical outputs. The following figure shows the sequence of On/Off controls generated by the drum pattern above. Compare the two, and you will find that they are equivalent! If you can see their equivalence, you are well on your way to understanding drum instruction operation.



Step Transitions

Drum Instruction Types

Drum instructions in the DL450 CPU consist of four types:

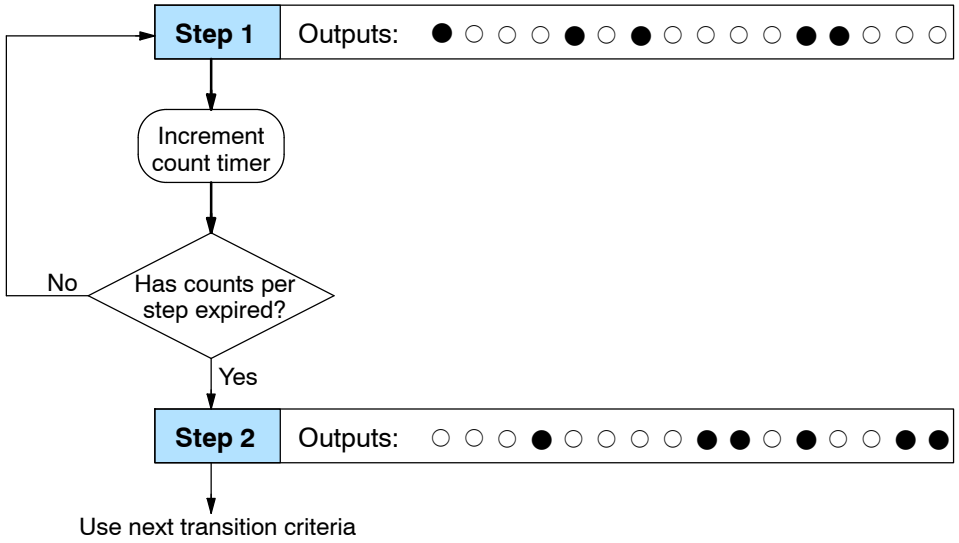
- Timed Drum with Discrete Outputs (DRUM)
- Time and Event Drum with Discrete Outputs (EDRUM)
- Masked Event Drum with Discrete Outputs (MDRMD)
- Masked Event Drum with Word Output (MDRMW)

The four drum instructions include time-based step transitions, and three include event-based transitions as well. Other options include outputs defined as a single word or as individual bits, and an output mask (individual output disable/enable).

Each drum has 16 steps, and each step has 16 outputs. Refering to the figure below, each output can be either an X, Y, or C coil, offering programming flexibility. Step 1 has been assigned an arbitrary, unique, output pattern (○= Off, ●= On) as shown.

Timer-Only Transitions

Drums move from one step to another based on time and/or an external event (input). Each step has its own transition condition which you assign during the drum instruction entry. The figure below shows how timer-only transitions work.



The drum remains in Step 1 for a specific duration (user-programmable). The timebase of the timer is programmable, from 0.01 seconds to 99.99 seconds. This establishes the resolution, or the duration of each “tick of the clock”. Each step uses the same timebase, but has its own unique counts per step, which you program. The drum spends a specific amount of time in each step, given by the formula:

Time in step = 0.01 seconds X Timebase x Counts per step

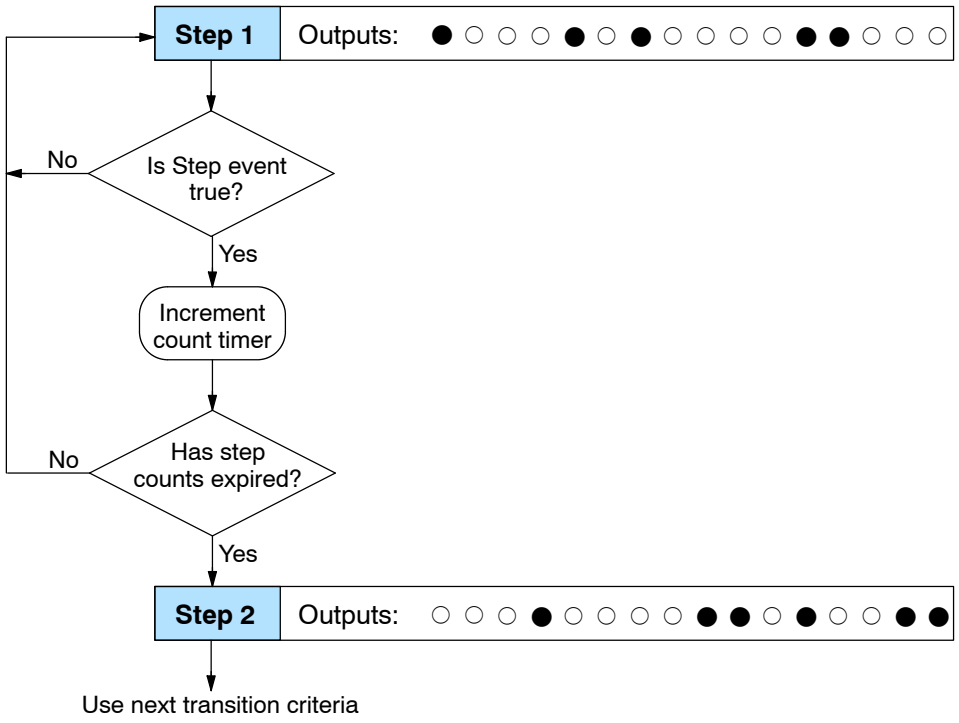
For example, if you program a 5 second time base and 12 counts for Step 1, then the drum will spend 60 seconds in Step 1. The maximum time for any step is given by the formula:

Max Time per step = 0.01 seconds X 9999 X 9999
= 999,800 seconds = 277.7 hours = 11.6 days

NOTE: When first choosing the timebase resolution, a good rule of thumb is to make it about 1/10 the duration of the shortest step in your drum. Then you will be able to optimize the duration of that step in 10% increments. Other steps with longer durations allow optimizing by even smaller increments (percentage-wise). Also, note that the drum instruction executes once per CPU scan. Therefore, it is pointless to specify a drum timebase that is much faster than the CPU scan time.

Timer and Event Transitions

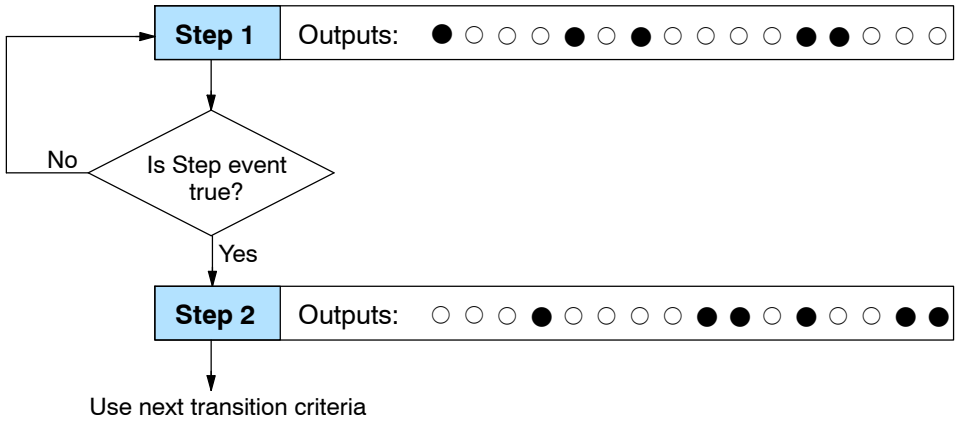
Time and Event Drums move from step to step based on time and/or external events. The figure below shows how step transitions work for these drums.



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 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 to be programmed for each step. You have the option of programming just one of the two, and even mixing transition types among all the steps of the drum. For example, you might want Step 1 to transition on an event, Step 2 to transition on time only, and Step 3 to transition on both time and an event. Furthermore, you may elect to use only part of the 16 steps, and only part of the 16 outputs.



Counter Assignments

Each drum instruction uses the resources of four counters in the CPU. When programming the drum instruction, you select the first counter number. The drum also uses the next three counters automatically. The counter bit associated with the first counter turns on when the drum has completed its cycle, going off when the drum is reset. These counter values and counter bit precisely indicate the progress of the drum instruction, and can be monitored by your ladder program.

Suppose we program a timer drum to have 8 steps, and we select CT10 for the counter number (remember, counter numbering is in octal). Counter usage is shown to the right. The right column holds typical values, interpreted below.

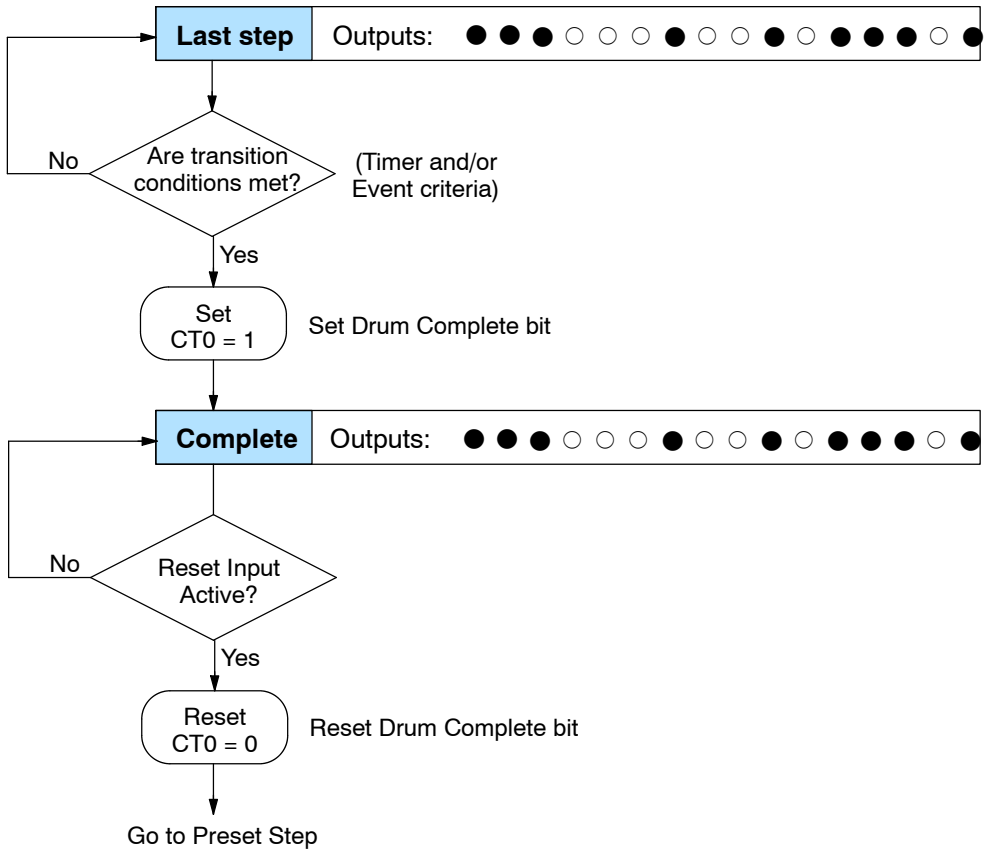
Counter Assignments

CT10	Counts in step	V1010	1528
CT11	Timer Value	V1011	0200
CT12	Preset Step	V1012	0001
CT13	Current Step	V1013	0004

CT10 shows that we are at the 1528th count in the current step, which is step 4 (shown in CT13). If we have programmed step 4 to have 3000 counts, then the step is just over half completed. CT11 is the count timer, shown in units of 0.01 seconds. So, each least-significant-digit change represents 0.01 seconds. The value of 200 means that we have been in the current count (1528) for 2 seconds (0.01 x 100). 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 does not change without a program edit. Counter bit CT10 turns on when the drum cycle is complete, and turns off when the drum is reset.

Last Step Completion

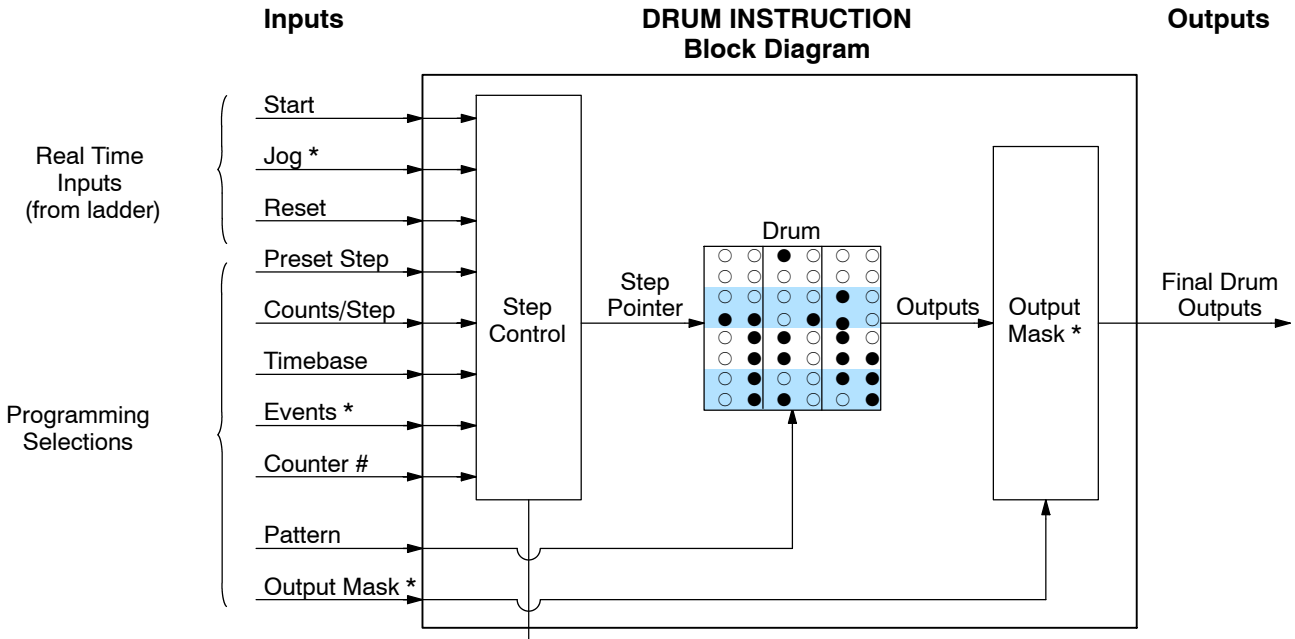
The last step in a drum sequence may be any step number, since partial drums are valid. Refer to the following figure. When the transition conditions of the last step are satisfied, the drum sets the counter bit corresponding to the counter named in the drum instruction box (such as CT0). Then it moves to a final “drum complete” state. The drum outputs remain in the pattern defined for the last step (including any output mask logic). 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 CT0), and then goes directly to the appropriate step number defined as the preset step.



Overview of Drum Operation

Drum Instruction Block Diagram

The drum instruction utilizes various inputs and outputs in addition to the drum pattern itself. Refer to the figure below.



Counter Assignments

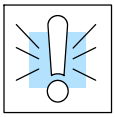
CT0	Counts in step	V1000	xxxx
CT1	Timer Value	V1001	xxxx
CT2	Preset Step	V1002	xxxx
CT3	Current Step	V1003	xxxx

* Asterisked inputs are applicable only to particular drum instructions.

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. Note that only the basic timer drum does not have a 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 on Timer/Event drums.
- **Timer Value** – the current value of the counts/step timer.
- **Counter #** – The counter number specifies the first of four consecutive counters which the drum uses for step control. You can monitor these to determine the drum's progress through its control cycle.
- **Events** – Either an X, Y, C, GX, GY, S, C, CT, or SP type discrete point serves as step transition inputs. Each step has its own event. However, programming the event is optional on Timer/Event drums.



WARNING: The outputs of a drum are enabled any time the CPU is in Run Mode. The Start Input *does not* have to be on, and the Reset input does not disable the outputs. Upon entering Run Mode, drum outputs automatically turn on or off according to the pattern of the preset step. This includes any effect of the output mask when applicable.

Powerup State of Drum Registers

The choice of the starting step on powerup and program-to-run mode transitions are important to consider for your application. Please refer to the following chart. If the counter memory is configured as non-retentive, the drum is initialized the same way on every powerup or program-to-run mode transition. However, if the counter memory is configured to be retentive, the drum will stay in its previous state.

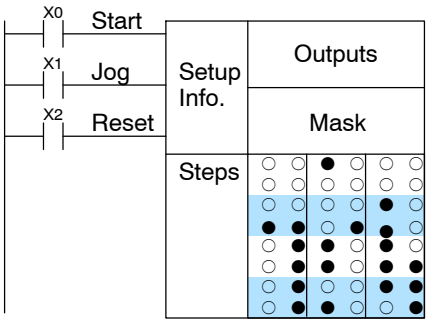
Counter Number	Function	Initialization on Powerup	
		Non-Retentive Case	Retentive Case
CTA(n)	Current Step Count	Initialize = 0	Use Previous (no change)
CTA(n + 1)	Counter Timer Value	Initialize = 0	Use Previous (no change)
CTA(n + 2)	Preset Step	Initialize = Preset Step #	Use Previous (no change)
CTA(n + 3)	Current Step #	Initialize = Preset Step #	Use Previous (no change)

Applications with relatively fast drum cycle times typically will need to be reset on powerup, using the non-retentive option. Applications with relatively long drum cycle times may need to resume at the previous point where operations stopped, using the retentive case. The default option is the retentive case. This means that if you initialize scratchpad V-memory, the memory will be retentive.

Drum Control Techniques

Drum Control Inputs

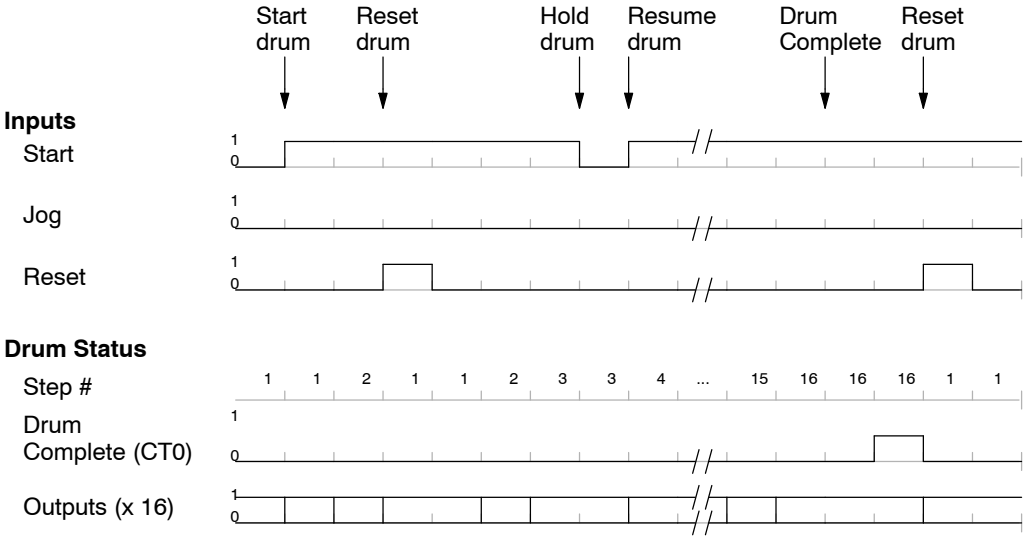
Now we are ready to put together the concepts on the previous pages and demonstrate general control of the drum instruction box. The drawing to the right shows a simplified generic drum instruction. Inputs from ladder logic control the Start, Jog, and Reset Inputs. The first counter bit of the drum (CT0, 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 is Step 1). When the Start input goes high the drum begins running, looking for an event and/or running the count timer (depending on the drum type and 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.

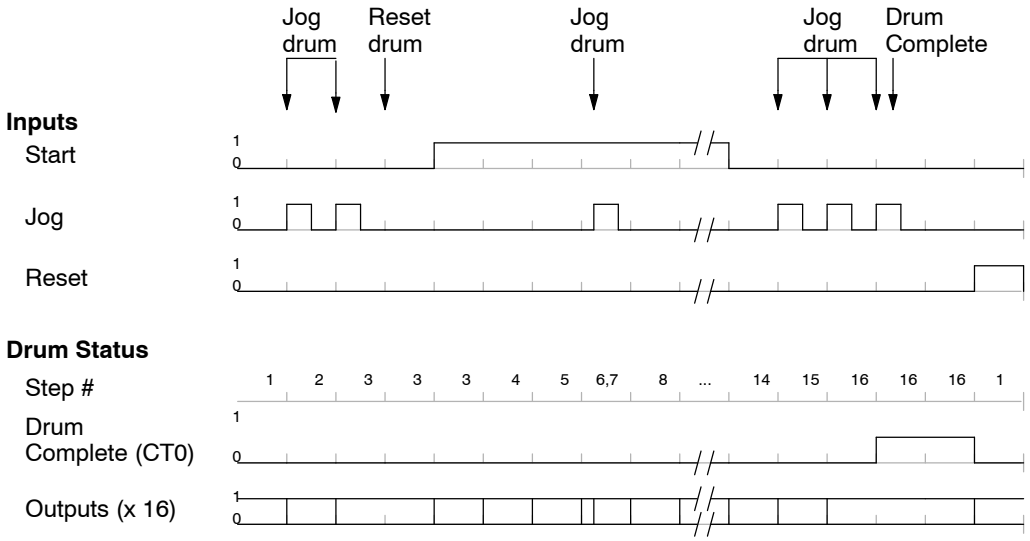


When the drum completes the last step (Step 16 in this example), the Drum Complete bit (CT0) turns on, and the step number remains at 16. When the Reset input turns on, it turns off the Drum Complete bit (CT0), and forces the drum to enter the preset step.

NOTE: The timing diagram shows all steps using equal time durations. Step times can vary greatly, depending on the counts/step programmed.

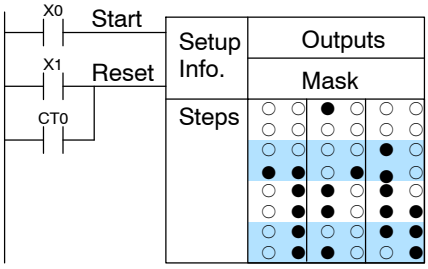
In the figure below, we focus on how the Jog input works on event drums. To the left of the diagram, note that the off-to-on transitions of the Jog input increments the step. Start may be either on or off (however, Reset must be off). Two jogs takes the drum to step three. Next, the Start input turns 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 CT0, so we logically OR the drum complete bit (CT0) with the Reset input. When the last step is done, the drum turns on CT0 which resets itself to the preset step, also resetting CT0. Contact X1 still works as a manual reset.



Initializing Drum Outputs

The outputs of a drum are enabled any time the CPU is in run mode. On program-to-run mode transitions, the drum goes to the preset step, and the outputs energize according to the pattern of that step. If your application requires all outputs to be off at powerup make the preset step in the drum a "reset step", with all outputs off.

Using Complex Event Step Transitions

Each event-based transition accepts only one contact reference for the event. However, this does not limit events to just one contact. Just use a control relay contact such as C0 for the step transition event. Elsewhere in the ladder logic you may use C0 as an output coil, making it dependent on many other "events" (contacts).

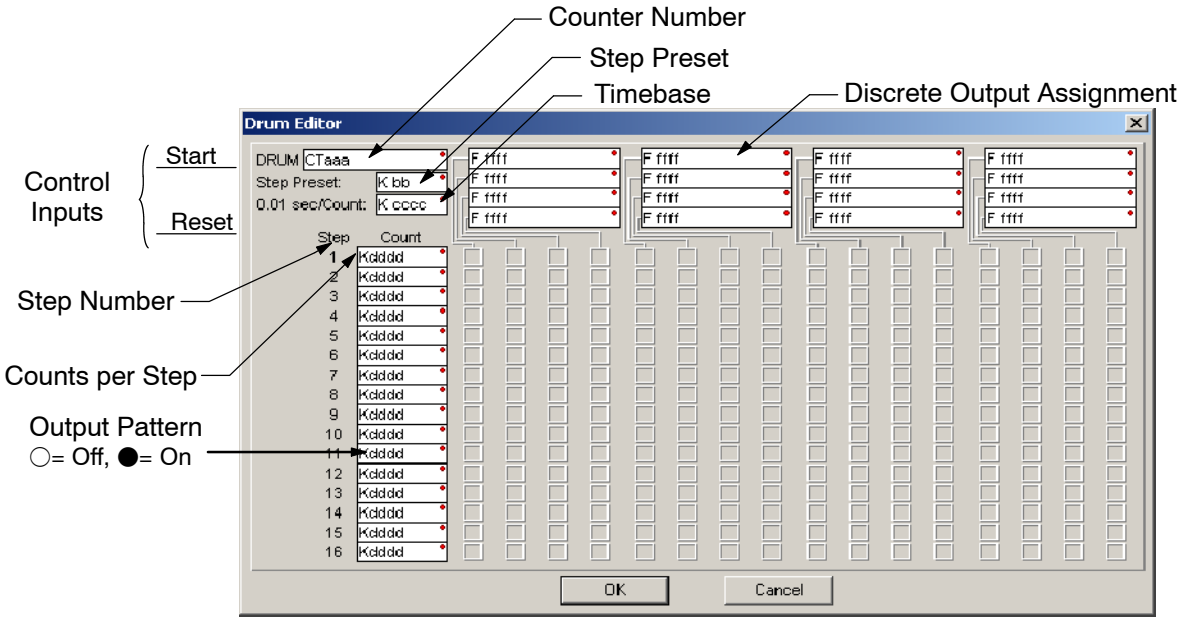
Drum Instructions

The DL450 drum instructions may be programmed ***Direct***SOFT. The EDRUM is the only drum instruction that can be programmed with a handheld programmer (firmware version 5.5 or later). This section covers entry using ***Direct***SOFT for all instructions plus the handheld mnemonics for the EDRUM instructions.

Timed Drum with Discrete Outputs (DRUM)

X	X	✓
430	440	450

The Timed Drum with Discrete Outputs is the most basic of the DL450's drum instructions. It operates according to the principles covered on the previous pages. Below is the instruction in chart form as displayed by ***Direct***SOFT.



The Timed Drum features 16 steps and 16 outputs. Step transitions occur only on a timed basis, specified in counts per step. Unused steps must be programmed with "counts per step" = 0 (this is the default entry). The discrete output points may be individually assigned as X, Y, or C types, or may be left unused. The output pattern may be edited graphically with ***Direct***SOFT.

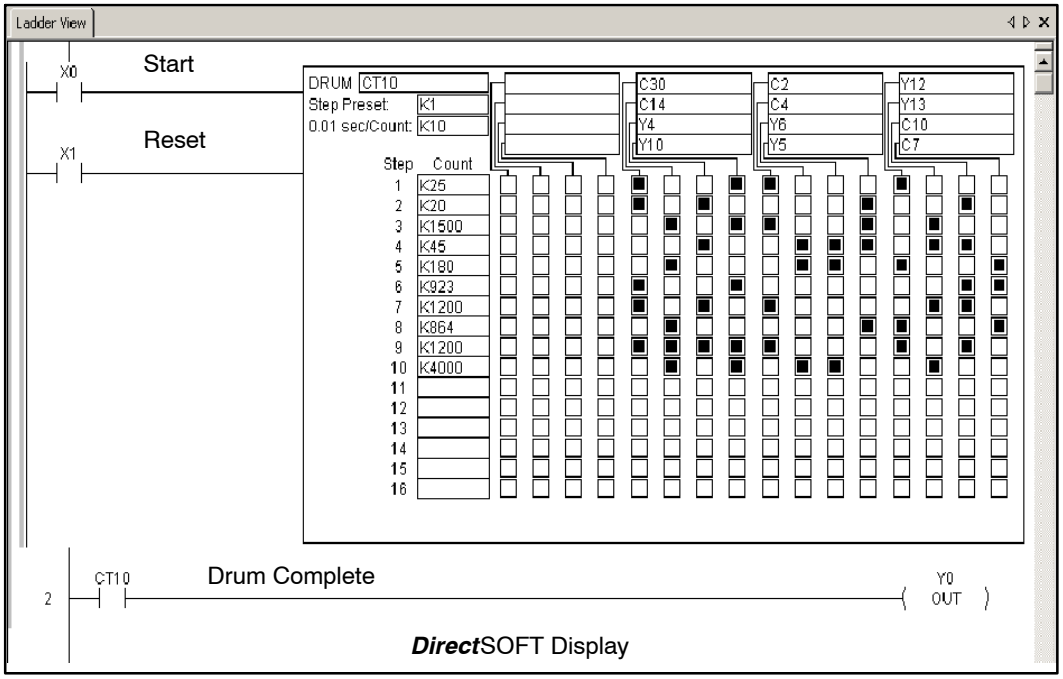
Whenever the Start input is energized, the drum's timer is enabled. It stops when the last step is complete, or when the Reset input is energized. The drum enters the preset step chosen upon a CPU program-to-run mode transition, and whenever the Reset input is energized.

Drum Parameters	Field	Data Types	Ranges
Counter Number	aaa	-	0 - 377
Preset Step	bb	K	1 - 16
Timer base	cc	K	0 - 99.99 seconds
Counts per step	dddd	K	0 - 9999
Discrete Outputs	Fffff	X, Y, C, GX, GY *	see page 3-42

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	CTn = 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+1) = (not used)

The following ladder program shows the DRUM instruction in a typical ladder program, as shown by ***Direct***SOFT. Steps 1 through 10 are used, and twelve of the sixteen output points are used. The preset step is step 1. The timebase runs at (K10 x 0.01) = 0.1 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.

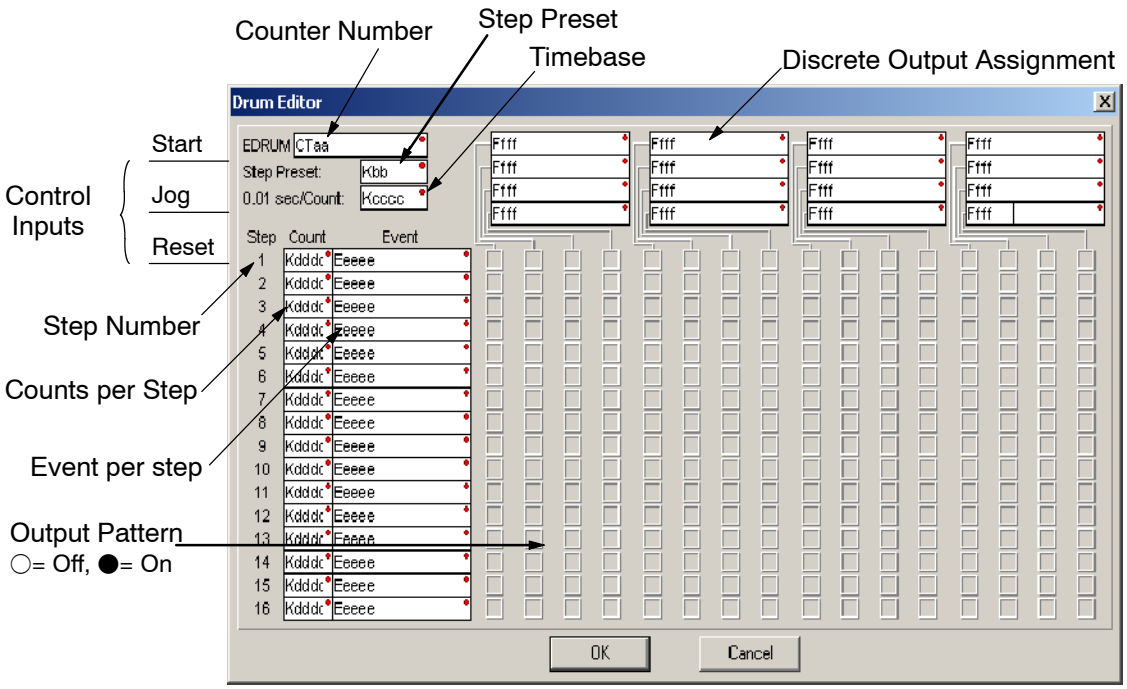


DirectSOFT Display

Event Drum (EDRUM)

X	X	✓
430	440	450

The Event Drum features time-based and event-based step transitions. It operates according to the general principles of drum operation covered in the beginning of this section. Below is the instruction as displayed by ***Direct***SOFT.



The Event Drum features 16 steps and 16 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 asssigned.

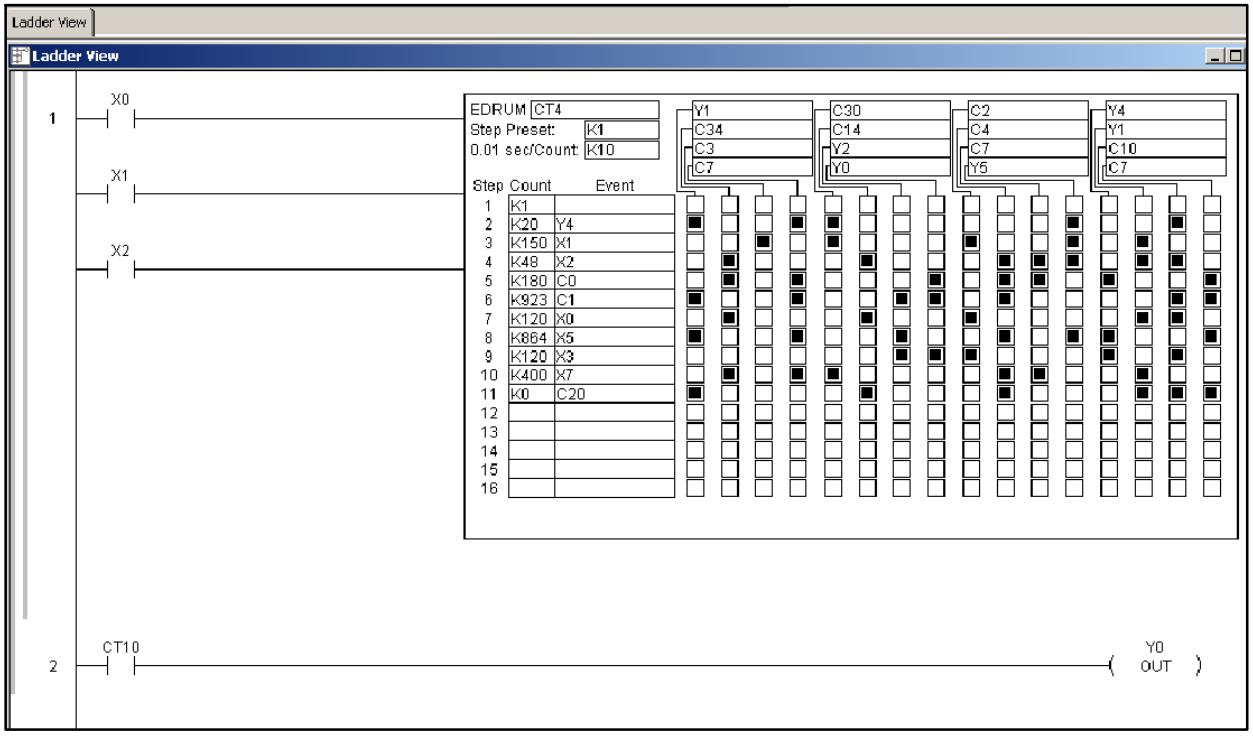
Drum Parameters	Field	Data Types	Ranges
Counter Number	aaa	-	0 - 377
Preset Step	bb	K	1 - 16
Timer base	cc	K	0 - 99.99 seconds
Counts per step	dddd	K	0 - 9999
Event	eeee	X, Y, C, GX, GY, S, T, ST	see page 3-42
Discrete Outputs	Fffff	X, Y, C , GX, GY*	see page 3-42

Whenever the Start input is energized, the drum's timer is enabled. As long as the event is true for the current step, the timer runs during that step. When the step count equals the counts per step, the drum transitions to the next step. This process stops when the last step is complete, or when the Reset input is energized. The drum enters the preset step chosen upon a CPU program-to-run mode transition, and whenever the Reset input is energized.

Drum instructions use four counters in the CPU. The ladder program can read the counter values for the drum's status. The ladder program may write a new preset step number to CTA(n+2) at any time. However, the other counters are for monitoring purposes only.

Counter Number	Ranges of (n)	Function	Counter Bit Function
CTA(n)	0 – 374	Counts in step	CTn = 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+1) = (not used)

The following ladder program shows the EDRUM instruction in a typical ladder program, as shown by ***Direct***SOFT. Steps 1 through 11 are used, and all 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 seconds. Note that step 1 is time-based only (event is left blank). And, the output pattern for step 1 programs all outputs off, which is a typically desirable powerup condition. In the last rung, the Drum Complete bit (CT4) turns on output Y0 upon completion of the last step (step 11). A drum reset also resets CT4.



NOTE: If all events are true in an event only drum (a drum with 0 counts per step in all steps), the PLC completes one step of the drum per scan; thus, the drum will be complete in 16 scans. However, as the outputs of the drum are enabled any time the CPU is in RUN Mode, the drum discrete outputs will be energized as pulsed outputs for each scan.

The handheld programmer can also enter or edit drum instructions. The diagram below lists the keystrokes for entering the drum example on the previous page.

NOTE: Drum editing requires Handheld Programmer firmware version 5.5 or later.

Handheld Programmer Keystrokes

Start	STR	X(IN)	0	↵			
Jog	STR	X(IN)	1	↵			
Reset	STR	X(IN)	2	↵			
Drum Inst.	SHFT	E	D	R	U	M	
	SHFT	CNT	4	↵			
Preset Step	(DEF K0000)	1	↵				
Time Base	(DEF K0000)	6	4	↵			
Outputs	0 (DEF 0000)	C(CR)	7	↵			
	(DEF 0000)	C(CR)	1	0	↵		
	(DEF 0000)	Y(OUT)	1	3	↵		
	(DEF 0000)	YOUT)	4	2	↵		
	(DEF 0000)	Y(OUT)	5	↵			
	(DEF 0000)	Y(OUT)	6	↵			
	(DEF 0000)	C(CR)	4	↵			
	(DEF 0000)	C(CR)	2	↵			
	(DEF 0000)	Y(OUT)	1	0	↵		
	(DEF 0000)	Y(OUT)	2	0	↵		
	(DEF 0000)	C(CR)	1	4	↵		
	(DEF 0000)	C(CR)	3	0	↵		
	(DEF 0000)	Y(OUT)	3	2	↵		
	(DEF 0000)	Y(OUT)	7	2	↵		
	(DEF 0000)	C(CR)	3	4	↵		
	15 (DEF 0000)	Y(OUT)	1	↵			
Counts/ Step	1 (DEF K0000)	5	↵				
	(DEF K0000)	2	0	↵			
	(DEF K0000)	1	5	0	↵		
	(DEF K0000)	4	5	↵			
	(DEF K0000)	1	8	0	↵		
	(DEF K0000)	9	2	3	↵		
	(DEF K0000)	1	2	0	↵		
	(DEF K0000)	8	6	4	↵		
	(DEF K0000)	1	2	0	0	↵	
	(DEF K0000)	4	0	0	↵		
	(DEF K0000)	NXT					
	(DEF K0000)	NXT					
	(DEF K0000)	NXT					
	(DEF K0000)	NXT					
	(DEF K0000)	NXT					
	16 (DEF K0000)	NXT					

← skip over unused steps

(Go to next column)

Handheld Programmer Keystrokes cont'd

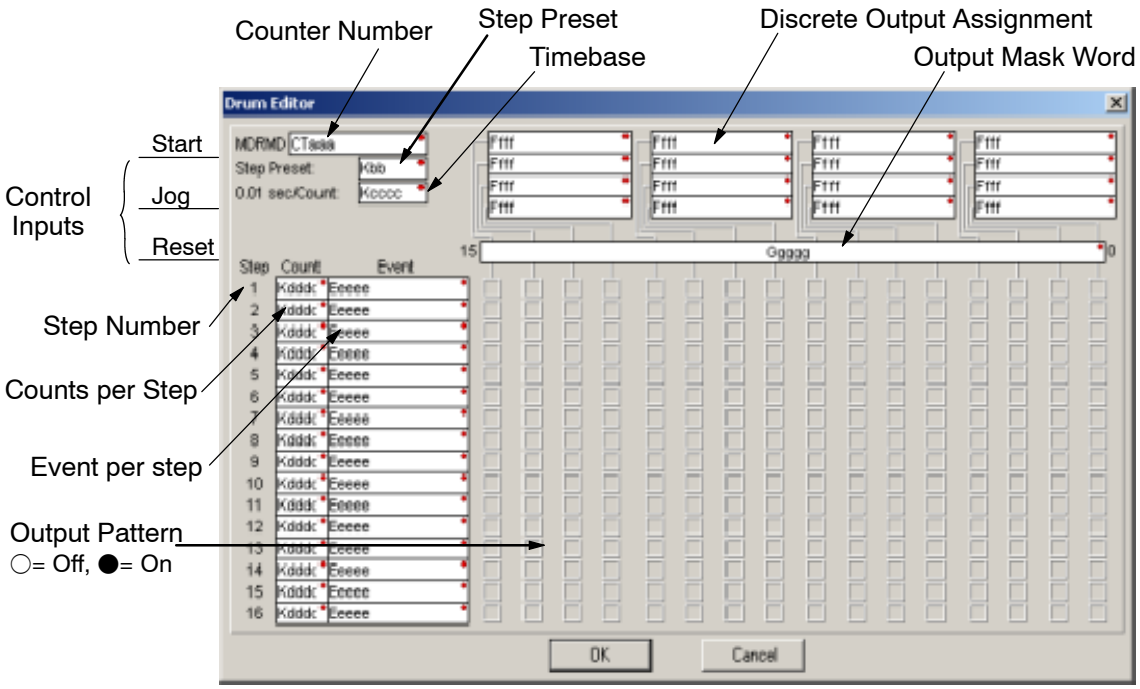
Events	1 (DEF 0000)	NXT	← skip over unused event			
	(DEF 0000)	Y(OUT)	4	0	↵	
	(DEF 0000)	X(IN)	2	1	↵	
	(DEF 0000)	X(IN)	2	2	↵	
	(DEF 0000)	C(CR)	0	↵		
	(DEF 0000)	C(CR)	1	↵		
	(DEF 0000)	X(IN)	3	0	↵	
	(DEF 0000)	X(IN)	3	5	↵	
	(DEF 0000)	X(IN)	3	3	↵	
	(DEF 0000)	Y(OUT)	1	7	↵	
	(DEF 0000)	C(CR)	2	0	↵	
	(DEF 0000)	NXT				
	(DEF 0000)	NXT				
	(DEF 0000)	NXT	← skip over unused steps			
	(DEF 0000)	NXT				
	16 (DEF 0000)	NXT				
Output Pattern	1 (DEF K0000)	NXT	← step 1 pattern = 0000			
	(DEF K0000)	9	8	1	2	↵
	(DEF K0000)	2	8	9	4	↵
	(DEF K0000)	4	4	7	6	↵
	(DEF K0000)	5	1	6	9	↵
	(DEF K0000)	9	3	4	3	↵
	(DEF K0000)	4	4	8	6	↵
	(DEF K0000)	9	4	5	9	↵
	(DEF K0000)	3	8	SHFT	A(H)	↵
	(DEF K0000)	5	8	6	4	↵
	(DEF K0000)	8	4	4	7	↵
	(DEF K0000)					
	(DEF K0000)	← unused steps				
	(DEF K0000)					
	(DEF K0000)					
	16 (DEF K0000)					
Last rung	STR	CNT	0	↵		
	Y(OUT)	0	↵			

NOTE: You may use the NXT and PREV keys to skip past entries for unused outputs or steps.

Masked Event Drum with Discrete Outputs (MDRMD)

X	X	✓
430	440	450

The Masked Event Drum with Discrete Outputs has all the features of the basic Event Drum plus final output control for each step. It operates according to the general principles of drum operation covered in the beginning of this section. Below is the instruction in chart form as displayed by ***Direct***SOFT.



The Masked Event Drum with Discrete Outputs features sixteen steps and sixteen outputs. Drum outputs are logically ANDed bit-by-bit with an output mask word for each step. The Gggg 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).

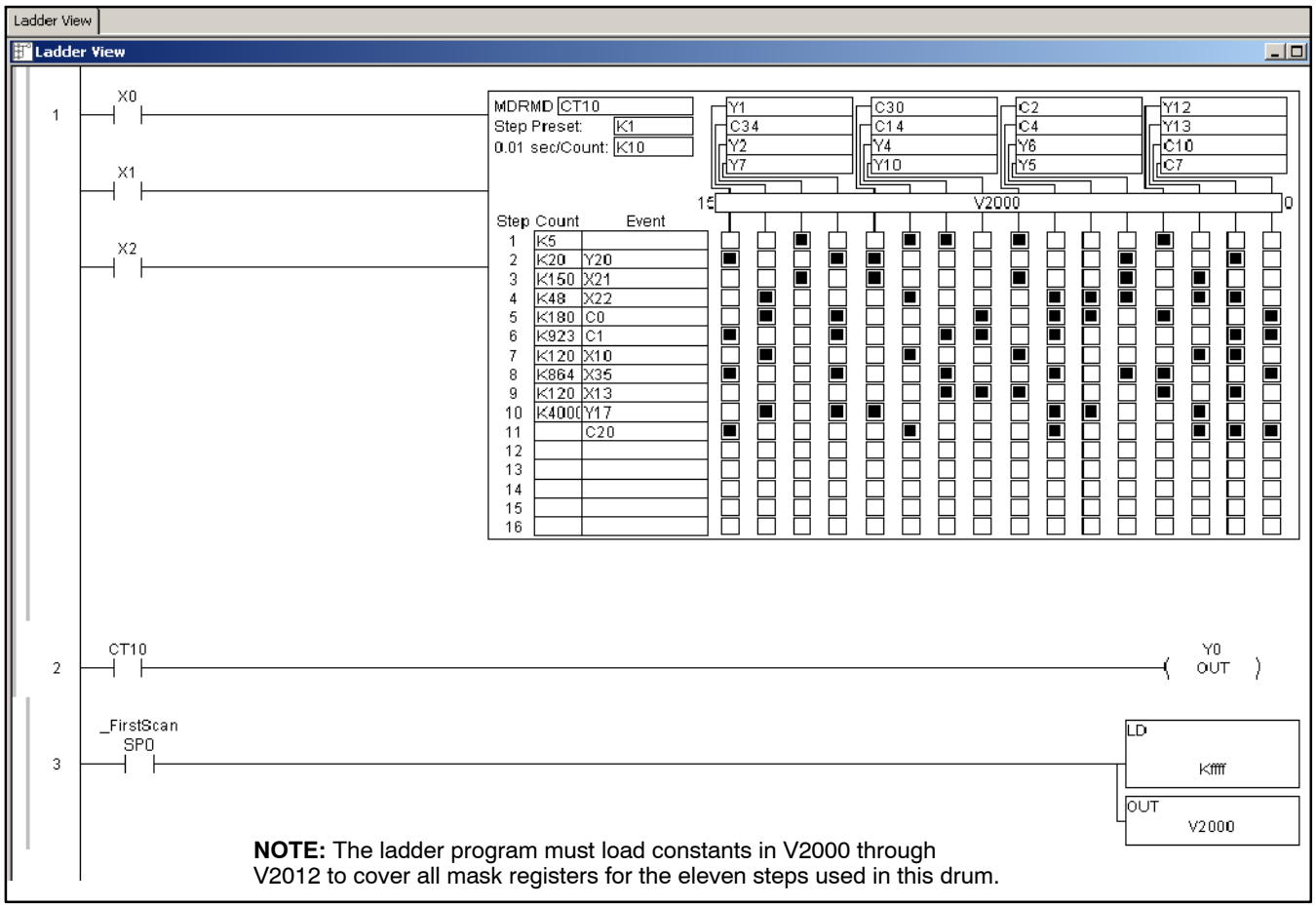
Drum Parameters	Field	Data Types	Ranges
Counter Number	aaa	-	0 - 177
Preset Step	bb	K	1 - 16
Timer base	cc	K	0 - 99.99 seconds
Counts per step	dddd	K	0 - 9999
Event	eeee	X, Y, C, GX, GY, S, T, ST	see page 3-42
Discrete Outputs	Fffff	X, Y, C, GX, GY *	see page 3-42
Output Mask	Gggg	V	see page 3-42

Whenever the Start input is energized, the drum's timer is enabled. As long as the event is true for the current step, the timer runs during that step. When the step count equals the counts per step, the drum transitions to the next step. This process stops when the last step is complete, or when the Reset input is energized. The drum enters the preset step chosen upon a CPU program-to-run mode transition, and whenever the Reset input is energized.

Drum instructions use four counters in the CPU. The ladder program can read the counter values for the drum's status. The ladder program may write a new preset step number to CTA(n+2) at any time. However, the other counters are for monitoring purposes only.

Counter Number	Ranges of (n)	Function	Counter Bit Function
CTA(n)	0 – 374	Counts in step	CTn = 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+1) = (not used)

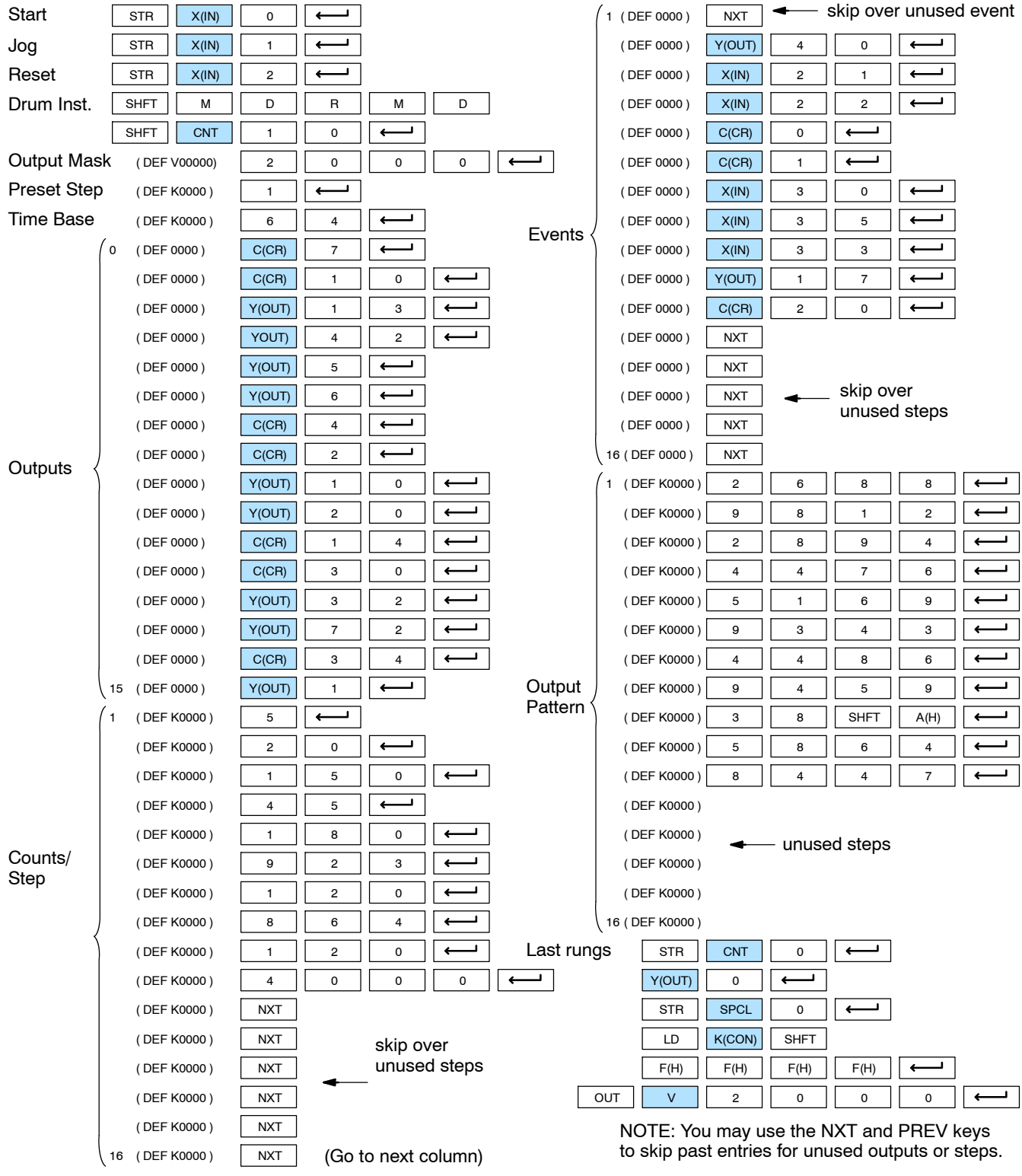
The following ladder program shows the MDRMD instruction in a typical ladder program, as shown by ***Direct***SOFT. Steps 1 through 11 are used, and all 16 output points are used. The output mask word is at V2000. The final drum outputs are shown above the mask word as individual bits. The data bits in V2000 are logically ANDed with the output pattern of the current step in the drum. If you want all drum outputs to be off after powerup, just 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.



NOTE: The ladder program must load constants in V2000 through V2012 to cover all mask registers for the eleven steps used in this drum.

The handheld programmer can also enter or edit drum instructions. The diagram below lists the keystrokes for entering the drum example on the previous page.

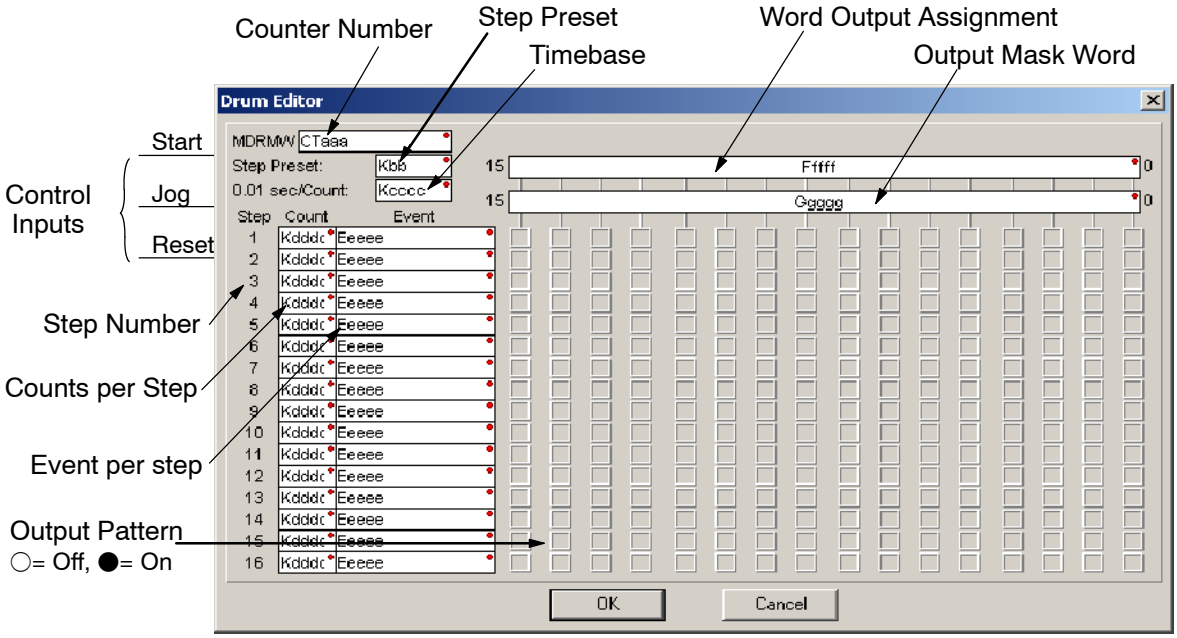
NOTE: Drum editing requires Handheld Programmer firmware version 5.5 or later.



Masked Event Drum with Word Output (MDRMW)

X	X	✓
430	440	450

The Masked Event Drum with Word Output features outputs organized as bits of a single word, rather than discrete points. It operates according to the general principles of drum operation covered in the beginning of this section. Below is the instruction in chart form as displayed by ***Direct***SOFT.



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 Gggg field specifies the beginning location of the 16 mask words, creating the final output (Ffff 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).

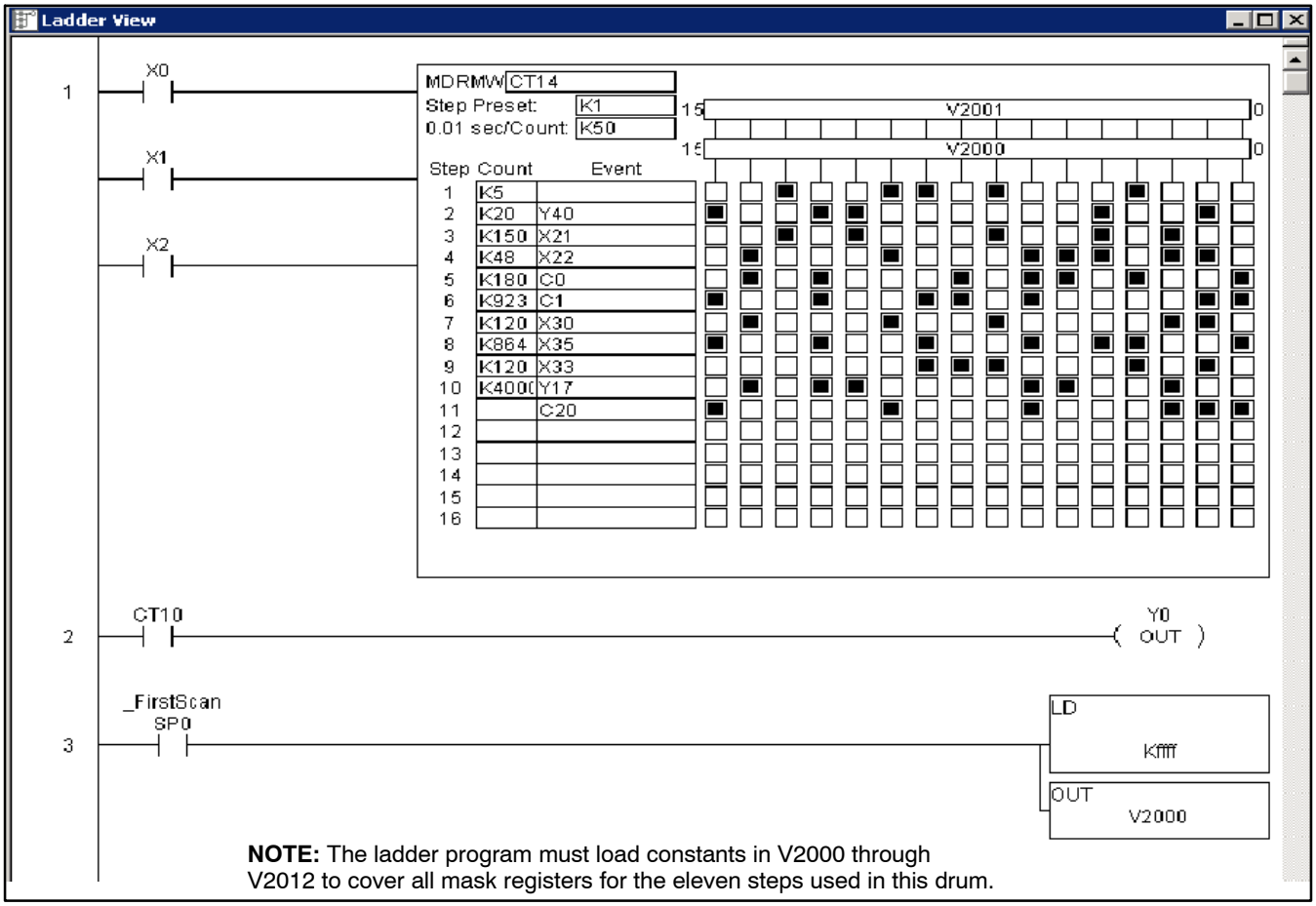
Drum Parameters	Field	Data Types	Ranges
Counter Number	aaa	-	0 - 177
Preset Step	bb	K	1 - 16
Timer base	cc	K	0 - 99.99 seconds
Counts per step	dddd	K	0 - 9999
Event	eeee	X, Y, C, GX, GY, S, T, ST	see page 3-42
Word Output	Ffffff	V	see page 3-42
Output Mask	Gggg	V	see page 3-42

Whenever the Start input is energized, the drum's timer is enabled. As long as the event is true for the current step, the timer runs during that step. When the step count equals the counts per step, the drum transitions to the next step. This process stops when the last step is complete, or when the Reset input is energized. The drum enters the preset step chosen upon a CPU program-to-run mode transition, and whenever the Reset input is energized.

Drum instructions use four counters in the CPU. The ladder program can read the counter values for the drum's status. The ladder program may write a new preset step number to CTA(n+2) at any time. However, the other counters are for monitoring purposes only.

Counter Number	Ranges of (n)	Function	Counter Bit Function
CTA(n)	0 – 374	Counts in step	CTn = 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+1) = (not used)

The following ladder program shows the MDRMD instruction in a typical ladder program, as shown by ***Direct***SOFT. Steps 1 through 11 are used, and all sixteen output points are used. The output mask word is at V2000. The final drum outputs are shown above the mask word as a word at V2001. The data bits in V2000 are logically ANDed with the output pattern of the current step in the drum, generating the contents of V2001. If you want all drum outputs to be off after powerup, write zeros to V2000 on the first scan. Ladder logic may update the output mask at any time to enable or disable the drum outputs. The preset step is step 1. The timebase runs at (K50 X 0.01) = 0.5 seconds per count. Therefore, the duration of step 1 is (5 x 0.5) = 2.5 seconds. Note that step 1 is time-based only (event is left blank). In the last rung, the Drum Complete bit (CT14) turns on output Y0 upon completion of the last step (step 10). A drum reset also resets CT14.



NOTE: The ladder program must load constants in V2000 through V2012 to cover all mask registers for the eleven steps used in this drum.

The handheld programmer can also enter or edit drum instructions. The diagram below lists the keystrokes for entering the drum example on the previous page.

NOTE: Drum editing requires Handheld Programmer firmware version 5.5 or later.

Handheld Programmer Keystrokes

Start	STR	X(IN)	0	↵				
Jog	STR	X(IN)	1	↵				
Reset	STR	X(IN)	2	↵				
Drum Inst.	SHFT	M	D	R	M	D		
	SHFT	CNT	1	4	↵			
Output Mask	(DEF V00000)	2	0	0	0	↵		
Preset Step	(DEF K0000)	1	↵					
Time Base	(DEF K0000)	6	4	↵				
Output Word	(DEF V00000)	2	0	0	1	↵		
Counts/ Step	1 (DEF K0000)	5	↵					
	(DEF K0000)	2	0	↵				
	(DEF K0000)	1	5	0	↵			
	(DEF K0000)	4	5	↵				
	(DEF K0000)	1	8	0	↵			
	(DEF K0000)	9	2	3	↵			
	(DEF K0000)	1	2	0	↵			
	(DEF K0000)	8	6	4	↵			
	(DEF K0000)	1	2	0	↵			
	(DEF K0000)	4	0	0	0	↵		
	(DEF K0000)	NXT						
	(DEF K0000)	NXT	← skip over unused steps					
	(DEF K0000)	NXT						
	(DEF K0000)	NXT						
	(DEF K0000)	NXT						
	16 (DEF K0000)	NXT	(Go to next column)					

Handheld Programmer Keystrokes cont'd

Events	1 (DEF 0000)	NXT	← skip over unused event			
	(DEF 0000)	Y(OUT)	4	0	↵	
	(DEF 0000)	X(IN)	2	1	↵	
	(DEF 0000)	X(IN)	2	2	↵	
	(DEF 0000)	C(CR)	0	↵		
	(DEF 0000)	C(CR)	1	↵		
	(DEF 0000)	X(IN)	3	0	↵	
	(DEF 0000)	X(IN)	3	5	↵	
	(DEF 0000)	X(IN)	3	3	↵	
	(DEF 0000)	Y(OUT)	1	7	↵	
	(DEF 0000)	C(CR)	2	0	↵	
	(DEF 0000)	NXT				
	(DEF 0000)	NXT				
	(DEF 0000)	NXT	← skip over unused steps			
	(DEF 0000)	NXT				
	16 (DEF 0000)	NXT				
Output Pattern	1 (DEF K0000)	2	6	8	8	↵
	(DEF K0000)	9	8	1	2	↵
	(DEF K0000)	2	8	9	4	↵
	(DEF K0000)	4	4	7	6	↵
	(DEF K0000)	5	1	6	9	↵
	(DEF K0000)	9	3	4	3	↵
	(DEF K0000)	4	4	8	6	↵
	(DEF K0000)	9	4	5	9	↵
	(DEF K0000)	3	8	SHFT	A(H)	↵
	(DEF K0000)	5	8	6	4	↵
	(DEF K0000)	8	4	4	7	↵
	(DEF K0000)					
	(DEF K0000)	← unused steps				
	(DEF K0000)					
	(DEF K0000)					
	16 (DEF K0000)					

Last rungs	STR	CNT	0	↵	
	Y(OUT)	0	↵		
	STR	SPCL	0	↵	
	LD	K(CON)	SHFT		
	F(H)	F(H)	F(H)	F(H)	↵
OUT	V	2	0	0	0 ↵

NOTE: You may use the NXT and PREV keys to skip past entries for unused outputs or steps.

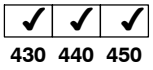
7

RLL*PLUS* Stage Programming

In This Chapter. . . .

- Introduction to Stage Programming
- Learning to Draw State Transition Diagrams
- Using the Stage Jump Instruction for State Transitions
- Stage Program Example: Toggle On/Off Lamp Controller
- Four Steps to Writing a Stage Program
- Stage Program Example: a Garage Door Opener
- Stage Program Design Considerations
- Parallel Processing Concepts
- Managing Large Programs
- RLL*PLUS* Instructions
- Questions and Answers About Stage Programming

Introduction to Stage Programming

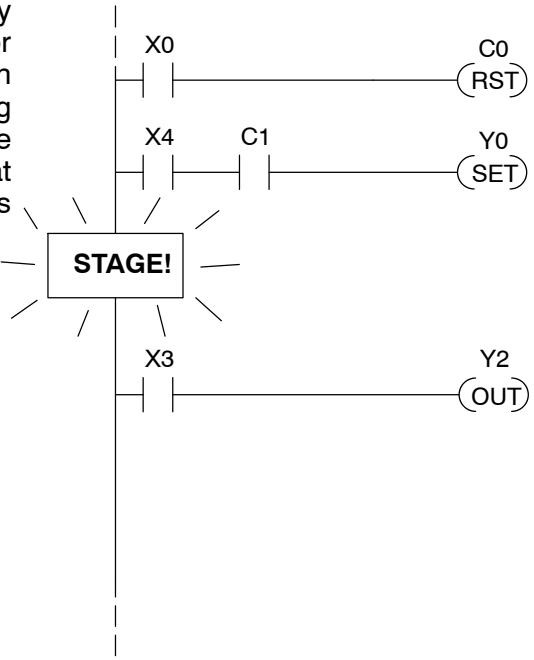


Stage Programming (available in all DL405 CPUs) provides a way to organize and program complex applications with relative ease, when compared to purely relay ladder logic (RLL) solutions. Stage programming does not replace or negate the use of traditional boolean ladder programming. This is why Stage Programming is also called RLL*PLUS*. You won't have to discard any training or experience you already have. Stage programming simply allows you to divide and organize a RLL program into groups of ladder instructions called stages. This allows quicker and more intuitive ladder program development than traditional RLL alone provides.

Overcoming "Stage Fright"

Many PLC programmers in the industry have become comfortable using RLL for every PLC program they write... but often remain skeptical or even fearful of learning new techniques such as stage programming. While RLL is great at solving boolean logic relationships, it has disadvantages as well:

- Large programs can become almost unmanageable, because of a lack of structure.
- In RLL, latches must be tediously created from self-latching relays.
- When a process gets stuck, it is difficult to find the rung where the error occurred.
- Programs become difficult to modify later, because they do not intuitively resemble the application problem they are solving.



It's easy to see that these inefficiencies consume a lot of additional time, and time is money. *Stage programming overcomes these obstacles!* We believe a few moments of studying the stage concept is one of the greatest investments in programming speed and efficiency a PLC programmer can make!

So, we encourage you to study stage programming and add it to your "toolbox" of programming techniques. This chapter is designed as a self-paced tutorial on stage programming. For best results:

- Start at the beginning and do not skip over any sections.
- Study each stage programing concept by working through each example. The examples build progressively on each other.
- Read the Stage Questions and Answers at the end of the chapter for a quick review.

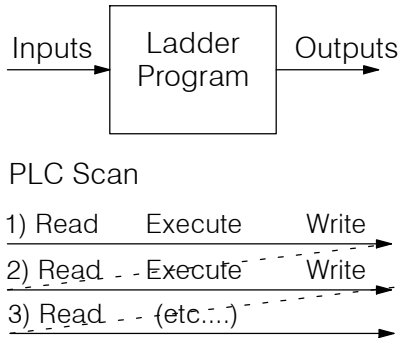
Learning to Draw State Transition Diagrams

Introduction to Process States

Those familiar with ladder program execution know 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.

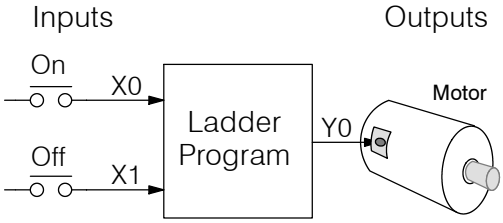
We can organize and divide ladder logic into sections called “stages”, representing process states. But before we describe stages in detail, we will reveal **the secret to understanding stage programming:** state transition diagrams.

The Need for State Diagrams

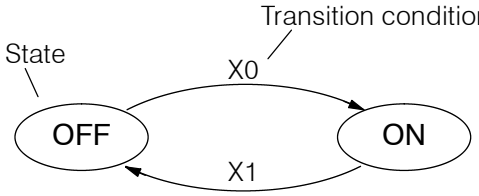
Sometimes we need to forget about the scan nature of PLCs, and focus our thinking toward the states of the process we need to identify. Clear thinking and concise analysis of an application gives us the best chance at writing efficient, bug-free programs. *State diagrams are just a tool to help us draw 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.

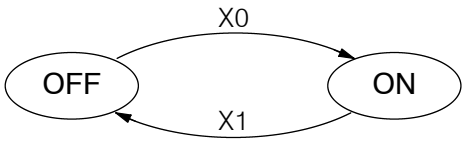


Output equation: Y0 = ON

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!*



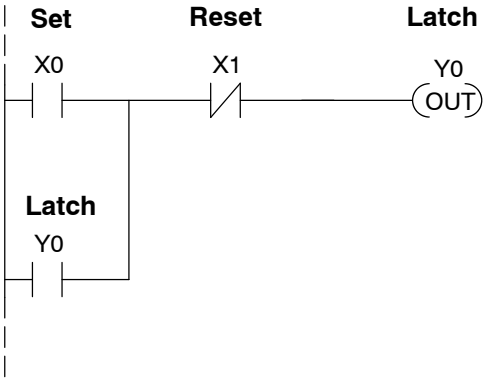
Output equation: Y0 = ON

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 Y0 has a dual purpose. When the On momentary pushbutton, X0, is pressed, output coil Y0 turns on and the Y0 contact on the second row latches itself on. So, X0 **sets the latch** Y0 on, and it remains on after the X0 contact opens. The output Y0 has power flow on a field device.

When the Off pushbutton (X1) is pressed, it opens the normally-closed X1 contact, which **resets the latch**, and output Y0 turns off.

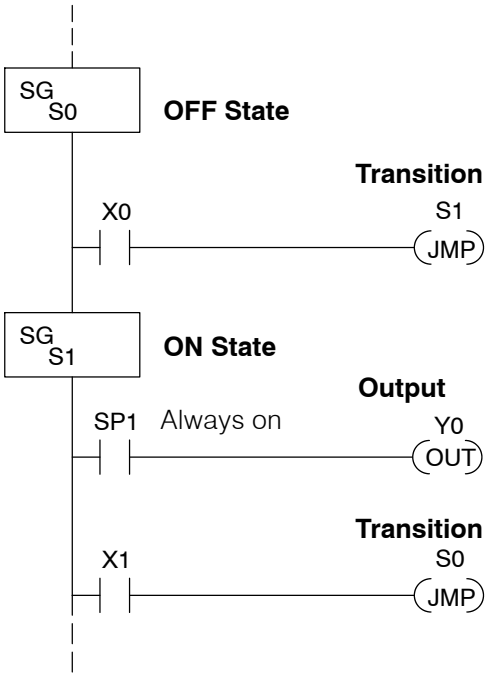


Stage Equivalent

The stage program solution is shown to the right. The two inline stage boxes S0 and S1 correspond to the two states OFF and ON. The ladder rung(s) below each stage box belong to each respective stage. This means that *the PLC only has to scan those rungs when the corresponding stage is active!*

For now, let's assume we begin in the OFF State, so stage S0 is active. When the On pushbutton (X0) is pressed, a stage transition occurs. The JMP S1 instruction executes, which simply turns off the Stage bit S0 and turns on Stage bit S1. So on the next PLC scan, the CPU will not execute Stage S0, but will execute stage S1!

In the On State (Stage S1), we want the motor to always be on. The special relay contact SP1 is defined as always on, so Y0 turns the motor on.

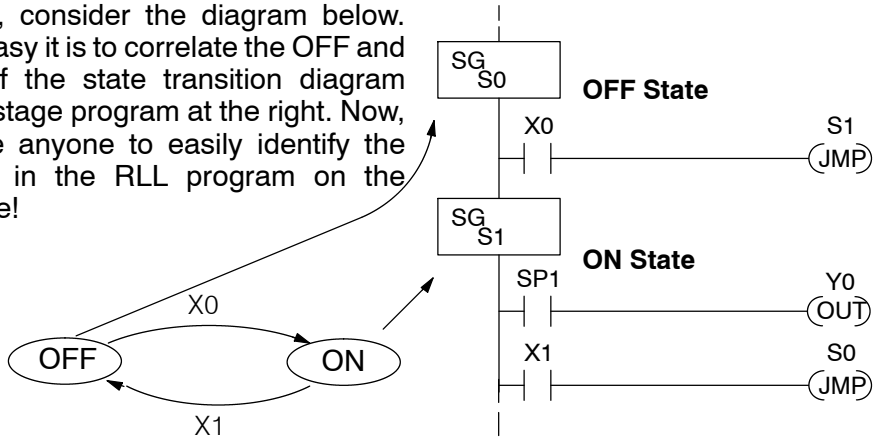


When the Off pushbutton (X1) is pressed, a transition back to the Off State occurs. The JMP S0 instruction executes, which simply turns off the Stage bit S1 and turns on Stage bit S0. On the next PLC scan, the CPU will not execute Stage S1, so the motor output Y0 will turn off. The Off state (Stage 0) will be ready for the next cycle.

Let's Compare

Right now, you may be thinking "I don't see the big advantage to Stage Programming... in fact, the stage program is longer than the plain RLL program". Well, now is the time to exercise a bit of faith. As control problems grow in complexity, stage programming quickly out-performs RLL in simplicity, program size, etc.

For example, consider the diagram below. Notice how easy it is to correlate the OFF and ON states of the state transition diagram below to the stage program at the right. Now, we challenge anyone to easily identify the same states in the RLL program on the previous page!

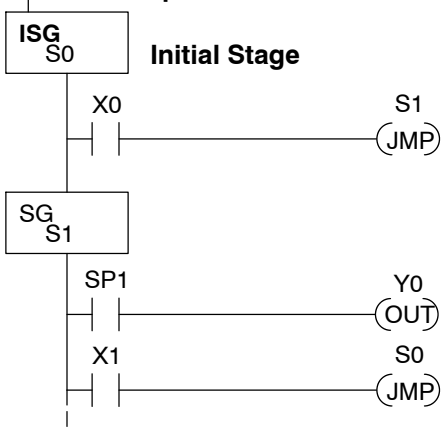


Initial Stages

At powerup and Program-to-Run Mode transitions, the PLC always begins with all normal stages (SG) off. So, the stage programs shown so far have actually had no way to get started (because rungs are not scanned unless their stage is active).

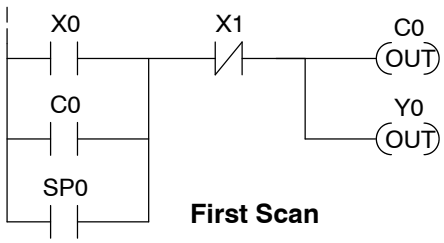
Assume that we want to always begin in the Off state (motor off), which is how the RLL program works. The Initial Stage (ISG) is defined to be active at powerup. In the modified program to the right, we have changed stage S0 to the ISG type. This ensures the PLC will scan contact X0 after powerup, because Stage S0 is active. **After powerup, an Initial Stage (ISG) works just like any other stage!**

Powerup in OFF State

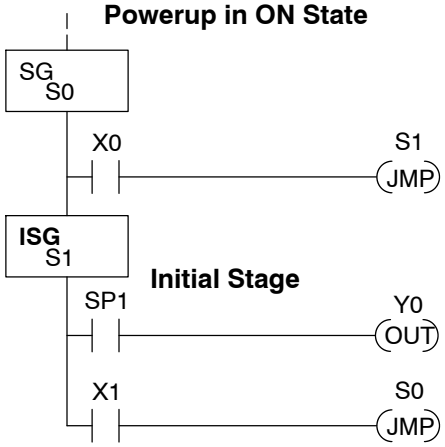


We can change both programs so that the motor is ON at powerup. In the RLL below, we must add a first scan relay SP0, latching C0 on. In the stage example to the right, we simply make Stage S1 an initial stage (ISG) instead of S0.

Powerup in ON State

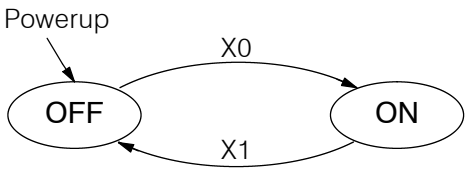


Powerup in ON State



NOTE: If the ISG is within the retentive range for stages, the ISG will remain in the state it was in before power down and will NOT turn itself on during the first scan.

We can mark our desired powerup state as shown to the right, which helps us remember to use the appropriate Initial Stages when creating a stage program. It is permissible to have as many initial stages as the process requires.



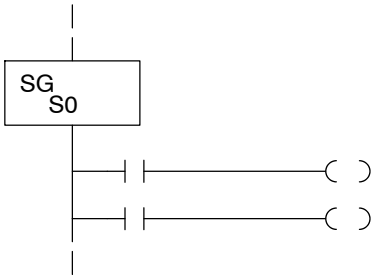
What Stage Bits Do

You may recall that a stage is just a section of ladder program which is either active or inactive at a given moment. All stage bits (S0 – Sxxx) reside in the PLC's image register as individual status bits. Each stage bit is either a boolean 0 or 1 at any time.

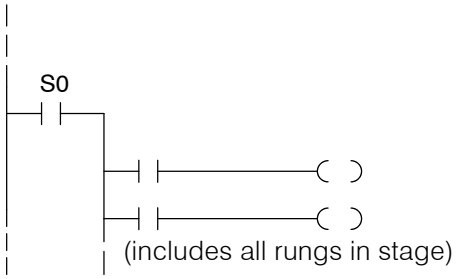
Program execution always reads ladder rungs from top to bottom, and from left to right. The drawing below shows the effect of stage bit status. The ladder rungs below the stage instruction continuing until the next stage instruction or the end of program belong to stage 0. Its equivalent operation is shown on the right. When S0 is true, the two rungs have power flow.

- If Stage bit S0 = 0, its ladder rungs *are not* scanned (executed).
- If Stage bit S0 = 1, its ladder rungs *are* scanned (executed).

Actual Program Appearance



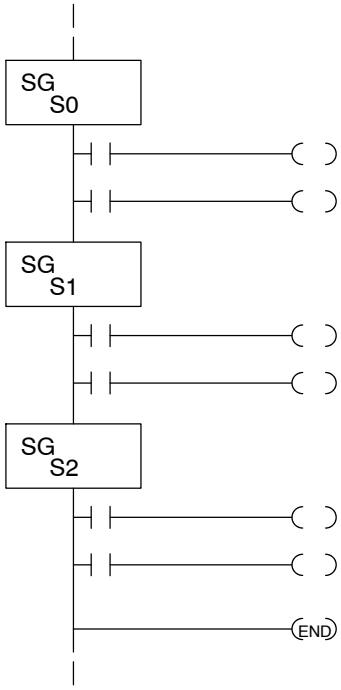
Functionally Equivalent Ladder



Stage Instruction Characteristics

The inline stage boxes on the left power rail divide the ladder program rungs into stages. Some stage rules are:

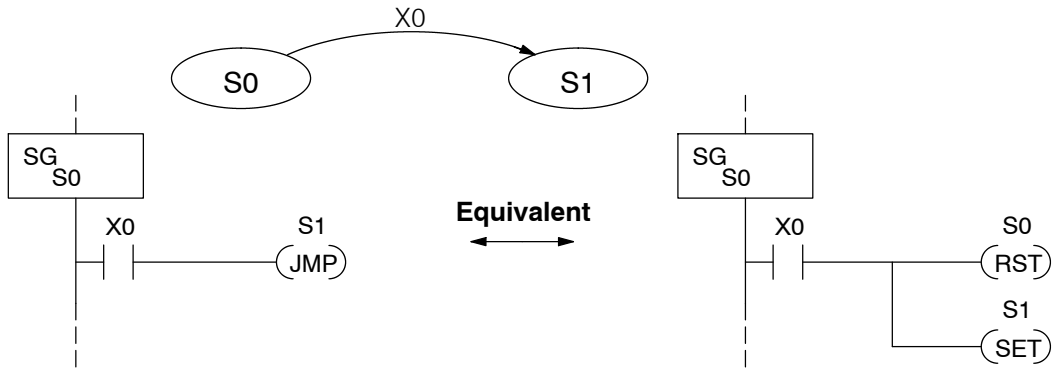
- **Execution** – Only logic in active stages are executed on any scan.
- **Transitions** – Stage transition instructions take effect on the next occurrence of the stages involved.
- **Octal numbering** – Stages are numbered in octal, like I/O points, etc. So "S8" is not valid.
- **Total Stages** – The maximum number of stages is CPU-dependent.
- **No duplicates** – Each stage number is unique and can be used just once.
- **Any order** – You can skip numbers and sequence the stage numbers in any order.
- **Last Stage** – the last stage in the ladder program includes all rungs from its stage box until the end coil.



Using the Stage Jump Instruction for State Transitions

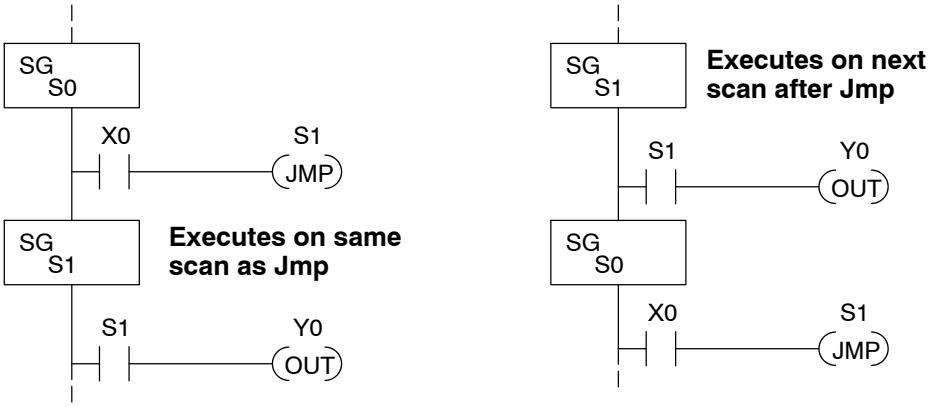
Stage Jump, Set, and Reset Instructions

The Stage JMP instruction we have used deactivates the stage in which the instruction occurs, while activating the stage in the JMP instruction. Refer to the state transition shown below. When contact X0 energizes, the state transition from S0 to S1 occurs. The two stage examples shown below are equivalent. So, the Stage Jump instruction is equal to a Stage Reset of the current stage, plus a Stage Set instruction for the stage to which we want to transition.



Please Read Carefully – The jump instruction is easily misunderstood. The "jump" does not occur immediately like a GOTO or GOSUB program control instruction when executed. Here's how it works:

- The jump instruction resets the stage bit of the stage in which it occurs. All rungs in the stage still finish executing during the current scan, *even if there are other rungs in the stage below the jump instruction!*
- The reset will be in effect on the following scan, so the stage that executed the jump instruction previously will be inactive and bypassed.
- The stage bit of the stage named in the Jump instruction will be set immediately, so the stage will be executed on its next occurrence. In the left program shown below, stage S1 executes during the *same scan* as the JMP S1 occurs in S0. In the example on the right, Stage S1 executes on the *next scan* after the JMP S1 executes, because stage S1 is located *above* stage S0.

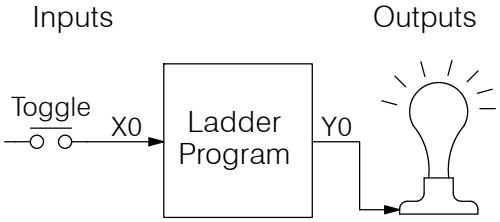


NOTE: Assume we start with Stage 0 active and Stage 1 inactive for both examples.

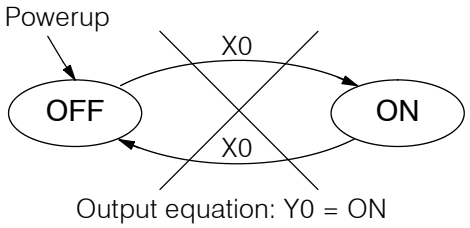
Stage Program Example: Toggle On/Off Lamp Controller

A 4-State Process

In the process shown to the right, we use an ordinary momentary pushbutton to control a light bulb. The ladder program will latch the switch input, so that we will push and release to turn on the light, push and release again to turn it off (sometimes called toggle function). Sure, we could just buy a mechanical switch with the alternate on/off action built in... However, this example is educational and also fun!

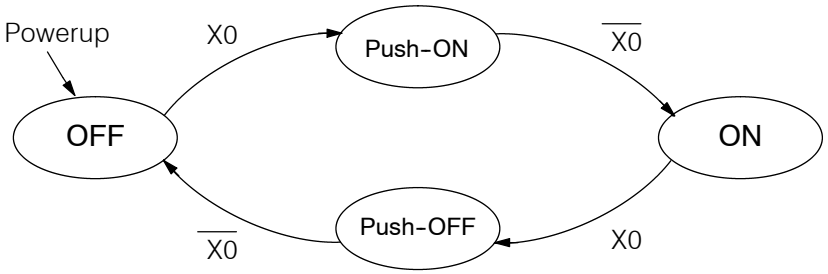


Next we draw the state transition diagram. A typical first approach is to use X0 for both transitions (like the example shown to the right). However, *this is incorrect* (please keep reading).



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)!

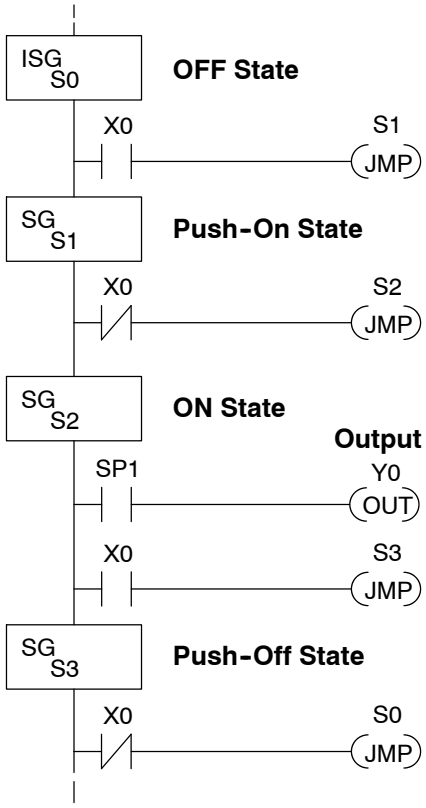
The solution is to make the push and the release of the pushbutton separate events. Refer to the new state transition diagram below. At powerup we enter the OFF state. When switch X0 is pressed, we enter the Press-ON state. When it is released, we enter the ON state. Note that X0 with the bar above it denotes X0 NOT.



When in the ON state, another push and release cycle similarly takes us back to the OFF state. Now we have two unique states (OFF and ON) used when the pushbutton is released, which is what was required to solve the control problem.

The equivalent stage program is shown to the right. The desired powerup state is OFF, so we make S0 an initial stage (ISG). In the ON state, we add special relay contact SP1, which is always on.

Note that even as our programs grow more complex, it is still easy to correlate the state transition diagram with the stage program!



Four Steps to Writing a Stage Program

By now, you've probably noticed that we follow the same steps to solve each example problem. The steps will probably come to you automatically if you work through all the examples in this chapter. It's helpful to have a checklist to guide us through the problem solving. The following steps summarize the stage program design procedure:

1. Write a Word Description of the application.

Describe all functions of the process in your own words. Start by listing what happens first, then next, etc. If you find there are too many things happening at once, try dividing the problem into more than one process. Remember, you can still have the processes communicate with each other to coordinate their overall activity.

2. Draw the Block Diagram.

Inputs represent all the information the process needs for decisions, and outputs connect to all devices controlled by the process.

- Make lists of inputs and outputs for the process.
- Assign I/O point numbers (X and Y) to physical inputs and outputs.

3. Draw the State Transition Diagram.

The state transition diagram describes the central function of the block diagram, reading inputs and generating outputs.

- Identify and name the states of the process.
- Identify the event(s) required for each transition between states.
- Ensure the process has a way to re-start itself, or is cyclical.
- Choose the powerup state for your process.
- Write the output equations.

4. Write the Stage Program.

Translate the state transition diagram into a stage program.

- Make each state a stage. Remember to number stages in octal. Up to 384 total stages are available in the DL430 CPU, numbered 0 to 577 octal. Up to 1024 total stages are available in the DL440/DL450 CPUs, 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 powerup.
- 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!

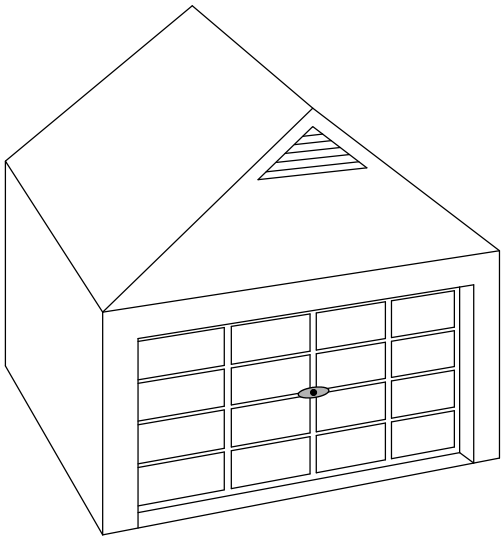
Stage Program Example: A Garage Door Opener

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 besides!

The first step we must take is to describe how the door opener works. We will start by achieving the basic operation, waiting to add extra features later (stage programs are very easy to modify).

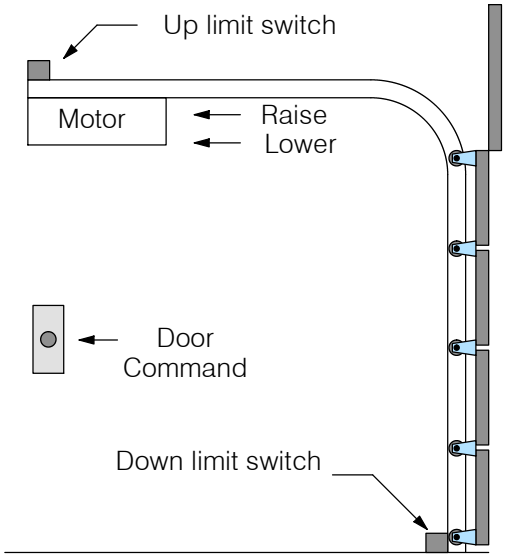
Our garage door controller has a motor which raises or lowers the door on command. The garage owner pushes and releases a momentary pushbutton once to raise the door. After the door is up, another push-release cycle will lower the door.



In order to identify the inputs and outputs of the system, it's sometimes helpful to sketch its main components, as shown in the door side view to the right. The door has an up limit and a down limit switch. Each limit switch closes only when the door has reached the end of travel in the corresponding direction. In the middle of travel, neither limit switch is closed.

The motor has two command inputs: raise and lower. When neither input is active, the motor is stopped.

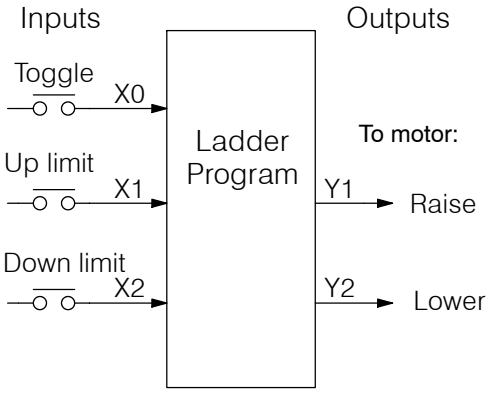
The door command is just a simple pushbutton. Whether wall-mounted as shown, or a radio-remote control, all door control commands logically OR together as one pair of switch contacts.



Draw the Block Diagram

The block diagram of the controller is shown to the right. Input X0 is from the pushbutton door control. Input X1 energizes when the door reaches the full up position. Input X2 energizes when the door reaches the full down position. When the door is positioned between fully up or down, both limit switches are open.

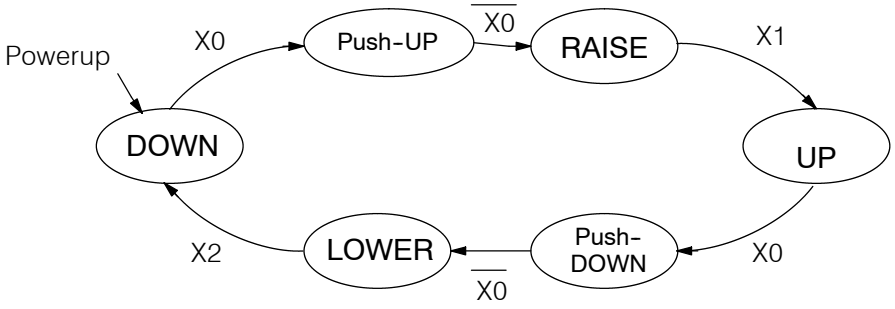
The controller has two outputs to drive the motor. Y1 is the up (raise the door) command, and Y2 is the down (lower the door) command.



Draw the State Diagram

Now we are ready to draw the state transition diagram. Like the previous light bulb controller example, this application also has 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.

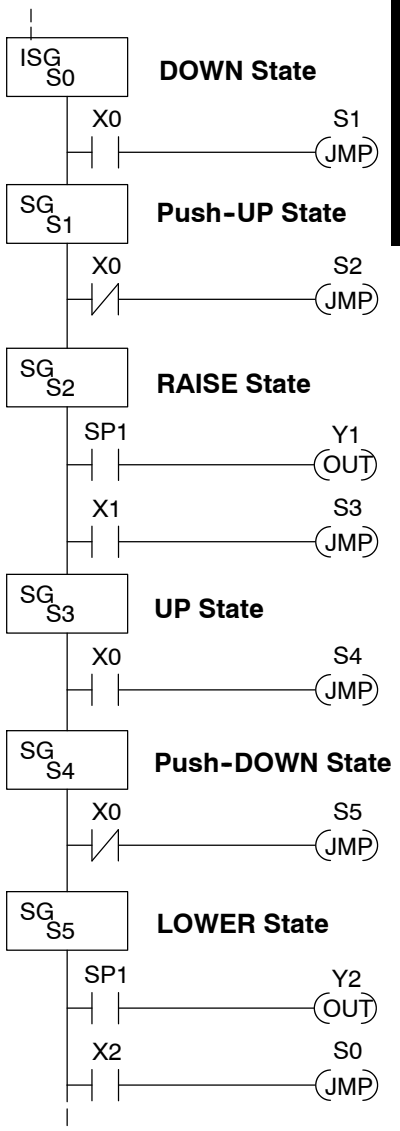


Output equations: Y1 = RAISE Y2 = LOWER

The equivalent stage program is shown to the right. For now, we will assume the door is down at powerup, so the desired powerup state is DOWN. We make S0 an initial stage (ISG). Stage S0 remains active until the door control pushbutton activates. Then we transition (JMP) to Push-UP stage, S1.

A push-release cycle of the pushbutton takes us through stage S1 to the RAISE stage, S2. We use the always-on contact SP1 to energize the motor's raise command, Y1. When the door reaches the fully-raised position, the up limit switch X1 activates. This takes us to the UP Stage S3, where we wait until another door control command occurs.

In the UP Stage S3, a push-release cycle of the pushbutton will take us to the LOWER Stage S5, where we activate Y2 to command the motor to lower the door. This continues until the door reaches the down limit switch, X2. When X2 closes, we transition from Stage S5 to the DOWN stage S0, where we began.



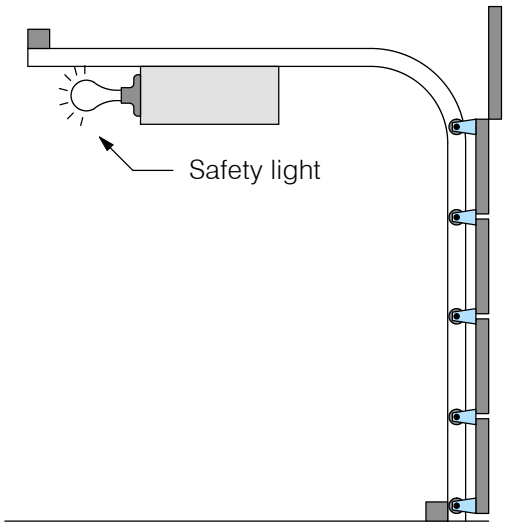
NOTE: The initial stage (ISG) is automatically active at powerup, afterwards, is acts like others.

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.

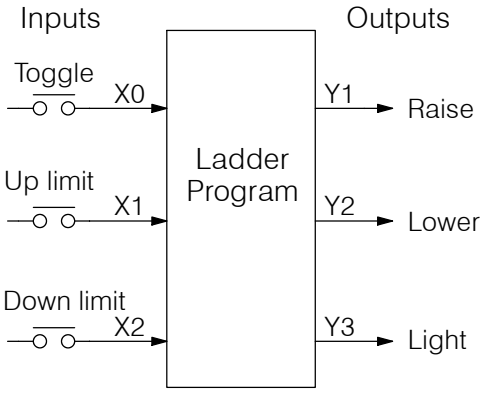
This part of the exercise will demonstrate the use of parallel states in our state diagram. Instead of using the JMP instruction, we'll use the set and reset commands.



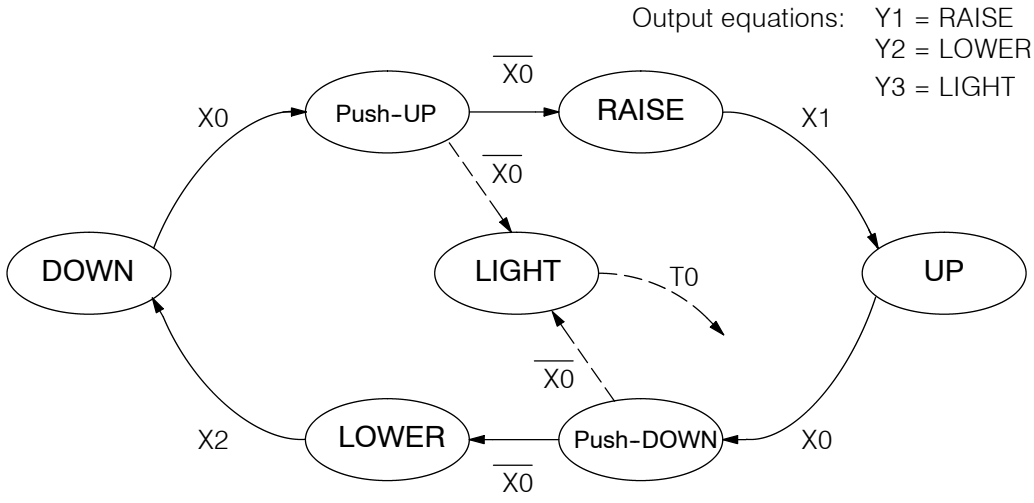
Modify the Block Diagram and State Diagram

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!



Using a Timer Inside a Stage

The finished modified program is shown to the right. The shaded areas indicate the program additions.

In the Push-UP stage S1, we add the Set Stage Bit S6 instruction. When contact X0 opens, we transition from S1 and go to two new active states: S2 and S6. In the Push-DOWN state S4, we make the same additions. So, any time someone presses the door control pushbutton, the light turns on.

Most new stage programmers would be concerned about where to place the Light Stage in the ladder, and how to number it. The good news is that it doesn't matter!

- 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.

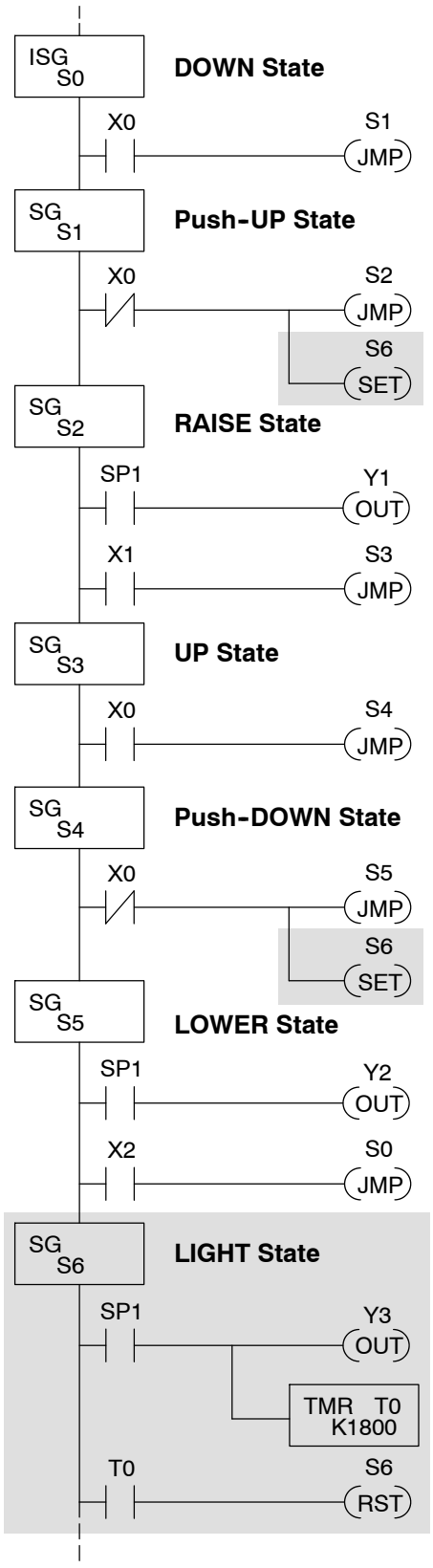
In stage S6, we turn on the safety light by energizing Y3. Special relay contact SP1 is always on. Timer T0 times at 0.1 second per count. To achieve 3 minutes time period, we calculate:

$$K = \frac{3 \text{ min.} \times 60 \text{ sec/min}}{0.1 \text{ sec/count}}$$

$$K = 1800 \text{ counts}$$

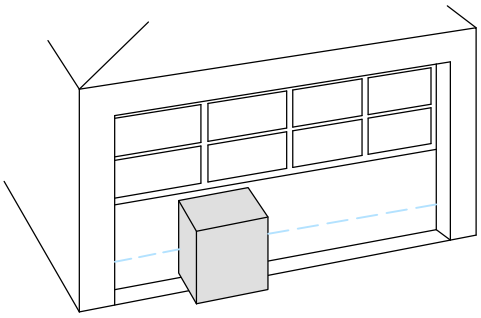
The timer has power flow whenever stage S6 is active. The corresponding timer bit T0 is set when the timer expires. So three minutes later, T0=1 and the instruction Reset S6 causes the stage to be inactive.

While Stage S6 is active and the light is on, stage transitions in the primary path continue normally and independently of Stage 6. That is, the door can go up, down, or whatever, but the light will be on for precisely 3 minutes.

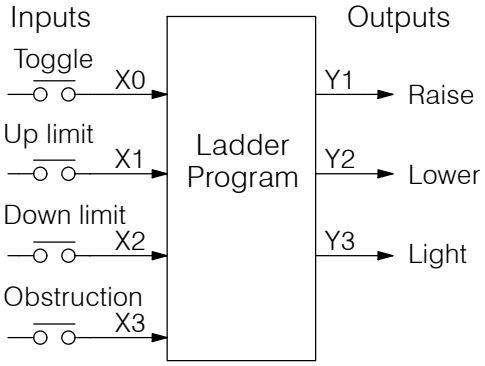


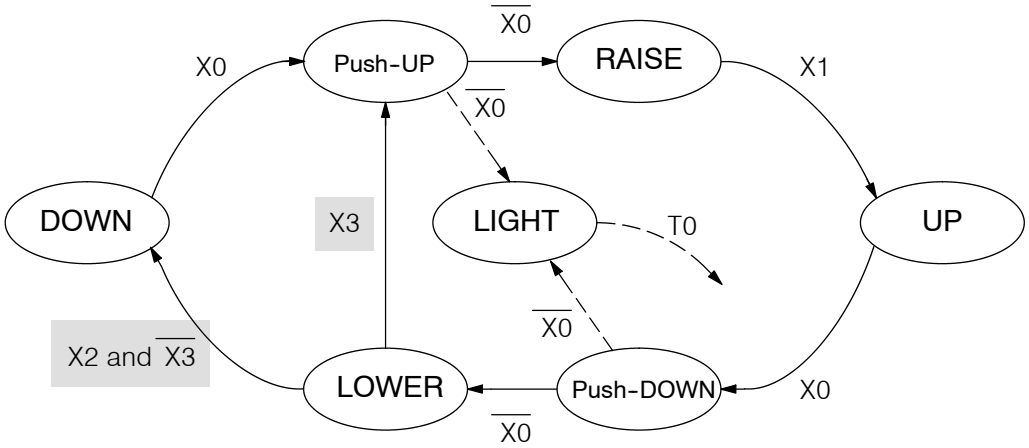
Add Emergency Stop Feature

Some garage door openers today will detect an object under the door. This halts further lowering of the door. Usually implemented with a photocell ("electric-eye"), a door in the process of being lowered will halt and begin raising. We will define our safety feature to work in this way, adding the input from the photocell to the block diagram as shown to the right. X3 will be on if an object is in the path of the door.



Next, we make a simple addition to the state transition diagram, shown in shaded areas in the figure below. Note the new transition path at the top of the LOWER state. If we are lowering the door and detect an obstruction (X3), we then jump to the Push-UP State. We do this instead of jumping directly to the RAISE state, to give the Lower output Y2 one scan to turn off, before the Raise output Y1 energizes.

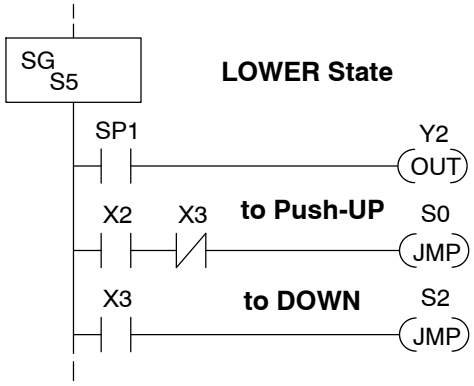




Exclusive Transitions

It is theoretically possible 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.

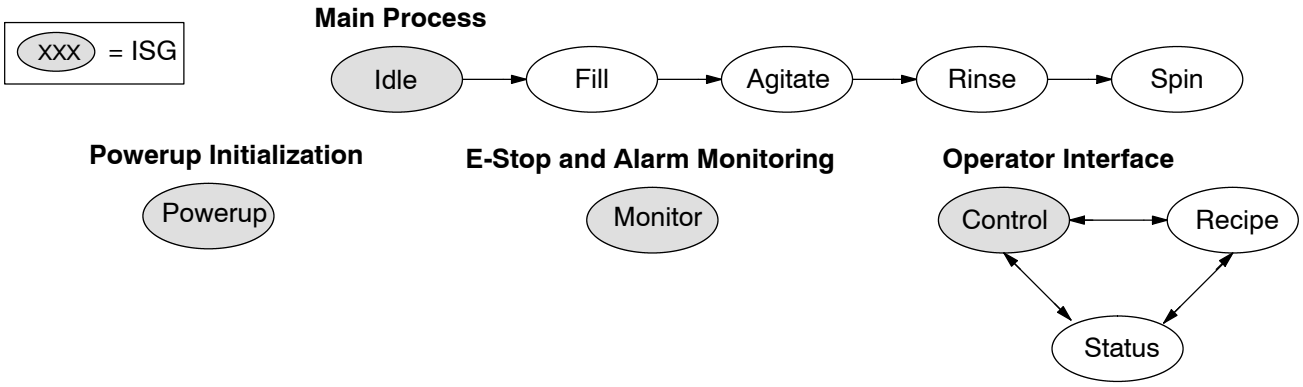


Stage Program Design Considerations

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.

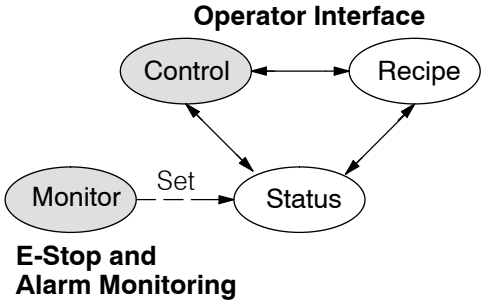
The following figure shows a typical application. During operation, the primary manufacturing activity Main Process, Powerup Initialization, E-Stop and Alarm Monitoring, and Operator Interface are all running. At powerup, four initial stages shown begin operation.



In a typical application, the separate stage sequences above operate as follows:

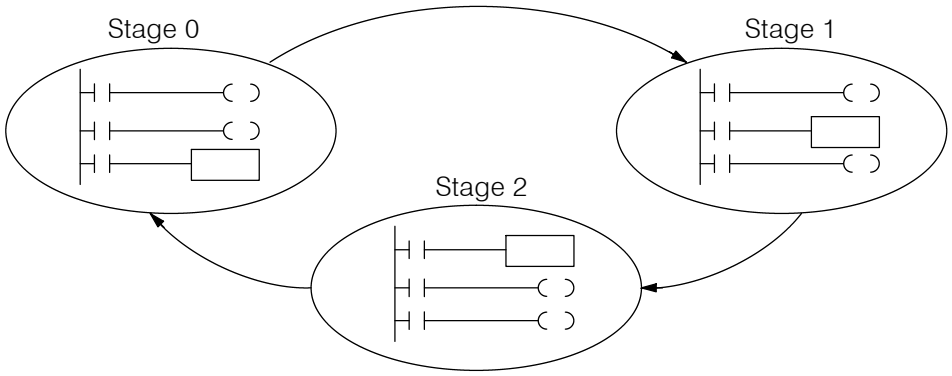
- **Powerup Initialization** – This stage contains ladder rung tasks done just once at powerup. Its last rung resets the stage, so this stage is only active for one scan (or only as many scans that are required).
- **Main Process** – this stage sequence controls the heart of the process or machine. One pass through the sequence represents one part cycle of the machine, or one batch in the process.
- **E-Stop and Alarm Monitoring** – This stage is always active because it is watching for errors that could indicate an alarm condition or require an emergency stop. It is common for this stage to reset stages in the main process or elsewhere, in order to initialize them after an error condition.
- **Operator Interface** – this is another task that must always be active and ready to respond to an operator. It allows an operator interface to change modes, etc. 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 powerup 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.
- Output coils automatically turn off when leaving a stage. However, Set and Reset instructions are not "undone" when leaving a stage.
- An output can be referenced from more than one stage, as long as only one of the stages is active at a time.
- If an output coil is controlled by more than one stage simultaneously, the active stage nearest the bottom of the program determines the final output status during each scan. So, use the OROUT instruction instead when you want multiple stages to have a logical OR control of an output.

One-Shot or PD coils – Use care if you must use a Positive Differential coil in a stage. Remember 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.

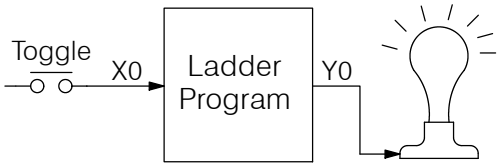
Counter – In using a counter inside a stage, the stage must be active for one scan before the input to the counter makes a 0–1 transition. Otherwise, there is no real transition and the counter will not count. The ordinary Counter instruction does have a restriction inside stages: it may not be reset from other stages using the RST instruction for the counter bit. However, the special Stage Counter provides a solution (see next paragraph).

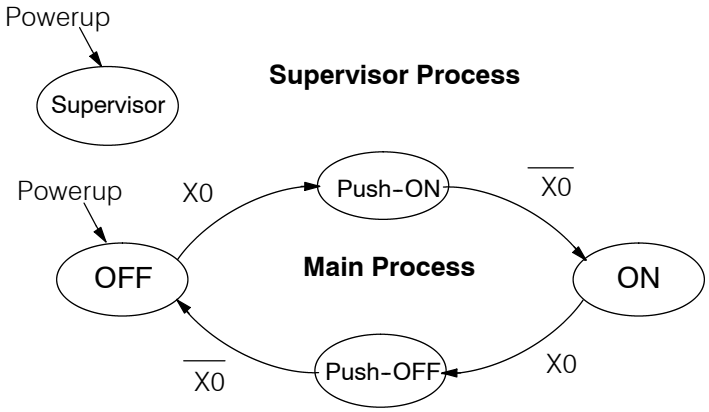
Stage Counter – The Stage Counter has the benefit that its count may be globally reset from other stages by using the RST instruction. It has a count input, but no reset input. This is the only difference from a standard counter instruction.

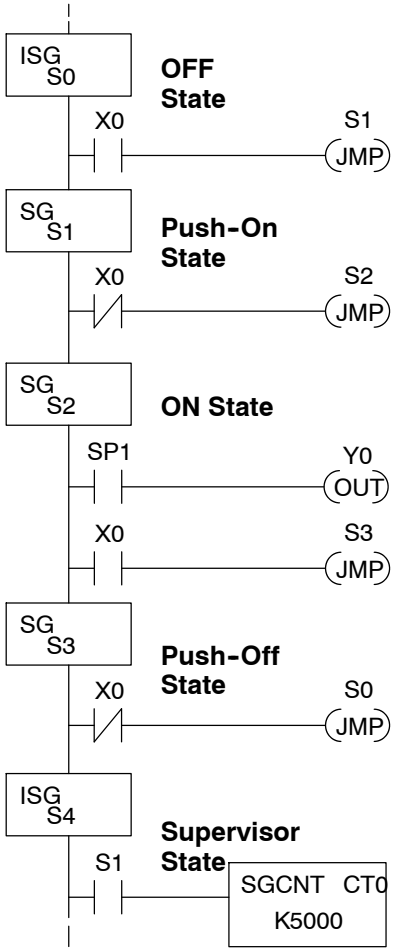
Drum – Realize that the drum sequencer is its own process, and is a different programming method than stage programming. If you need to use a drum and stages, be sure to place the drum instruction in an ISG stage that is always active.

Using a Stage as a Supervisory Process

You may recall the light bulb on-off controller example from earlier in this chapter. For the purpose of illustration, suppose we want to monitor the "productivity" of the lamp process, by counting the number of on-off cycles 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 the stages of the process they are trying to monitor. The problem with this approach is that the stage is active only part of the time. In order for the counter to count, the count input must transition from off to on at least one scan after its stage activates. Ensuring this requires extra logic that can be tricky.

In this case, we only need to add another supervisory stage as shown above, to "watch" the main process. The counter inside the supervisor stage uses the stage bit S1 of the main process as its count input. *Stage bits used as a contact let us monitor a process!*

NOTE: Both the Supervisor stage and the OFF stage are initial stages. The supervisor stage remains active indefinitely.

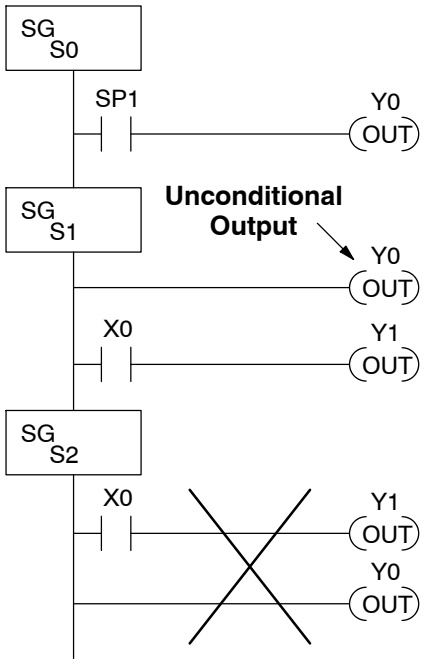
Stage Counter

The counter in the above example is a special Stage Counter. Note that it does not have a reset input. The count is reset by executing a Reset instruction, naming the counter bit (CT0 in this case). The Stage Counter has the benefit that its count may be globally reset from other stages. The standard Counter instruction does not have this global reset capability. You may still use a regular Counter instruction inside a stage... however, the reset input to the counter is the only way to reset it.

Unconditional Outputs

As in most example programs in this chapter and Stage 0 to the right, your application may require a particular output to be ON unconditionally when a particular stage is active. Until now, the examples always use the SP1 special relay contact (always on) in series with the output coils.

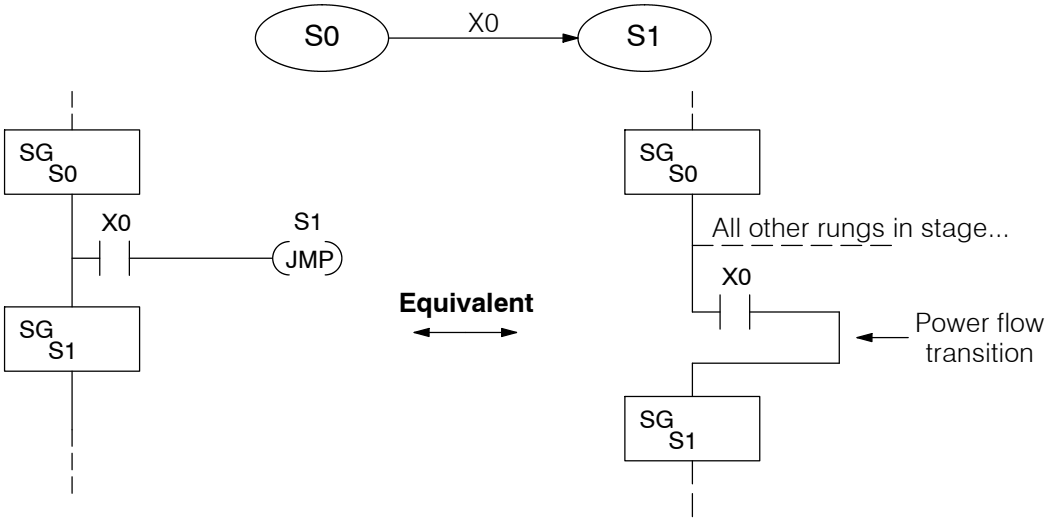
It's possible to omit the contact, as long as you place any unconditional outputs first (at the top) of a stage section of ladder. The first rung of Stage 1 does this.



WARNING: Unconditional outputs placed elsewhere in a stage do not necessarily remain on when the stage is active. In Stage 2 to the right, Y0 is shown as an unconditional output, but its power flow comes from the rung above. So, Y0 status will be the same as Y1 (is not correct).

Power Flow Transition Technique

Our discussion of state transitions has shown how the Stage JMP instruction makes the current stage inactive and the next stage (named in the JMP) active. As an alternative way to enter this in ***Direct***SOFT, you may use the power flow method for stage transitions. The main requirement is 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.



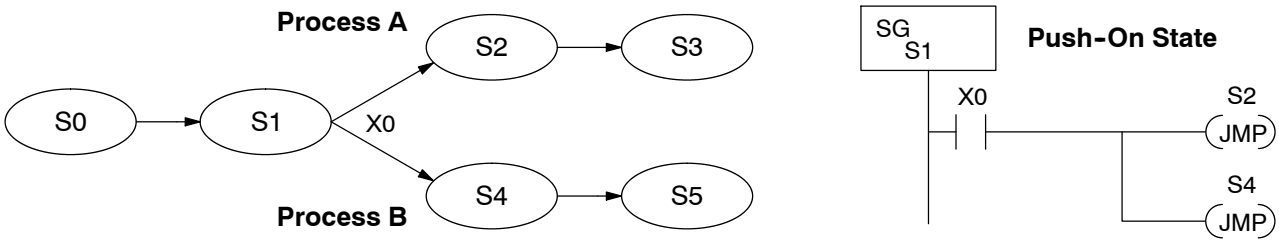
Recall that the Stage JMP instruction may occur anywhere in the current stage, and the result is the same. However, power flow transitions (shown above) must occur as the last rung in a stage. All other rungs in the stage will precede it. The power flow transition method is also achievable on the handheld programmer, by simply following the transition condition with the Stage instruction for the next stage.

The power flow transition method does eliminate one Stage JMP instruction, its only advantage. However, it is not as easy to make program changes as using the Stage JMP. Therefore, we advise using Stage JMP transitions for most programmers.

Parallel Processing Concepts

Parallel Processes

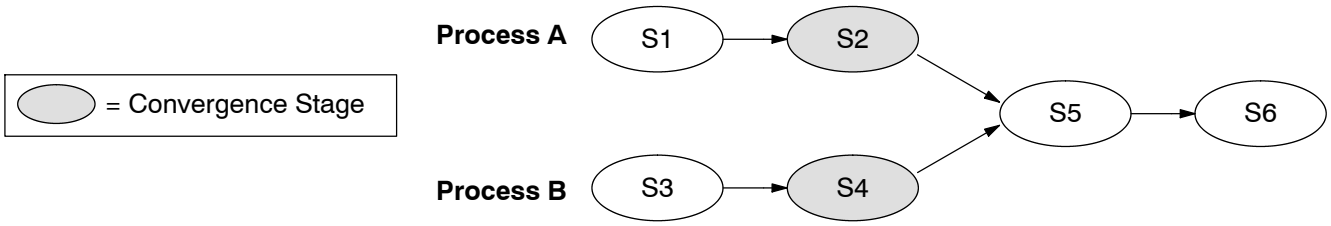
Previously in this chapter we discussed how a state may transition to either one state or another, called an *exclusive transition*. In other cases, we may need to branch simultaneously to two or more parallel processes, as shown below. It is acceptable to use all JMP instructions as shown, or we could 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 10–7). 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*.

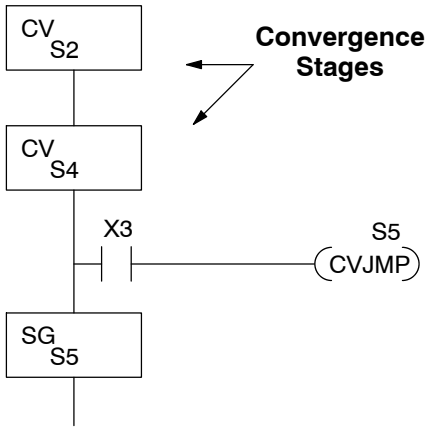


Convergence Stages (CV)

X	✓	✓
430	440	450

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.

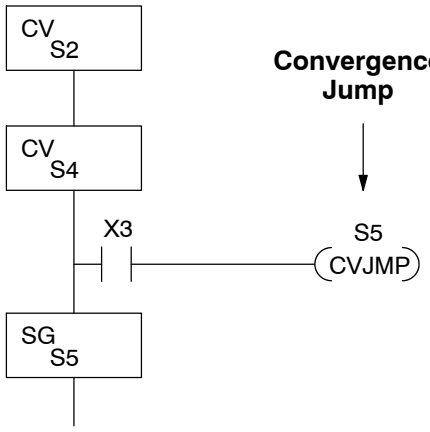
The solution is to coordinate the transition condition out of convergence stages. We accomplish this with a stage type designed for this purpose: the Convergence Stage (type CV). In the example to the right, convergence stages S2 and S4 are required to be grouped together as shown. **No logic is permitted 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)

X	✓	✓
430	440	450

Recall that the last convergence stage only has power flow when all CV stages in the group are active. To complement the convergence stage, we need a new jump instruction. The Convergence Jump (CVJMP) shown to the right will transition to Stage S5 when X3 is active (as one might expect), but it also *automatically resets all convergence stages in the group*. This makes the CVJMP jump a very powerful instruction. Note that this instruction may only be used with convergence stages.



Convergence Stage Guidelines

The following summarizes the requirements in the use of convergence stages, including some tips for their effective application:

- A convergence stage is to be used as the last stage of a process which is running in parallel to another process or processes. A transition to the convergence stage means that a particular process is through, and represents a waiting point until all other parallel processes also finish.
- The maximum number of convergence stages which make up one group is 17. In other words, a maximum of 17 stages can converge into one stage.
- Convergence stages of the same group must be placed together in the program, connected on the power rail without any other logic in between.
- Within a convergence group, the stages may occur in any order, top to bottom. It does not matter which stage is last in the group, because all convergence stages have to be active before the last stage has power flow.
- The last convergence stage of a group may have ladder logic within the stage. However, this logic will not execute until all convergence stages of the group are active.
- The convergence jump (CVJMP) is the intended method to be used to transition from the convergence group of stages to the next stage. The CVJMP resets all convergence stages of the group, and energizes the stage named in the jump.
- The CVJMP instruction must only be used in a convergence stage, as it is invalid in regular or initial stages.
- Convergence Stages or CVJMP instructions may not be used in subroutines or interrupt routines.

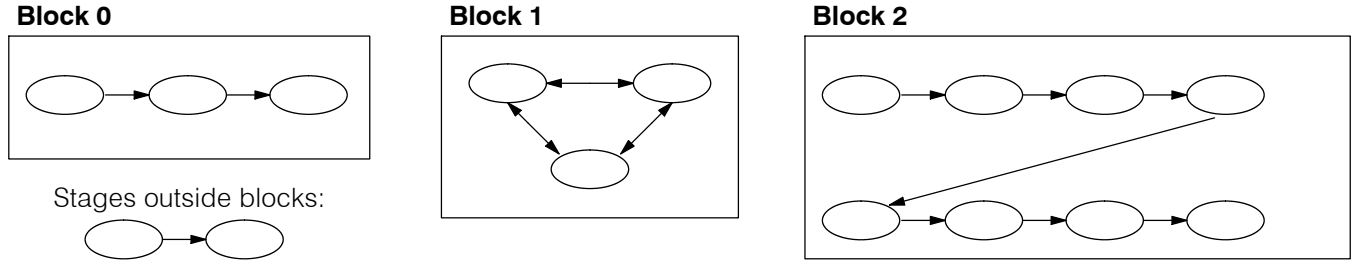
Managing Large Programs

A stage may contain a lot of ladder rungs, or just one or two program rungs. For most applications, good program design will ensure the average number of rungs per stage will be small. However, large application programs will still create a large number of *stages*. We introduce a new construct which will help us organize related stages into groups called *blocks*. So, program organization is the main benefit of the use of stage blocks.

Stage Blocks (BLK, BEND)

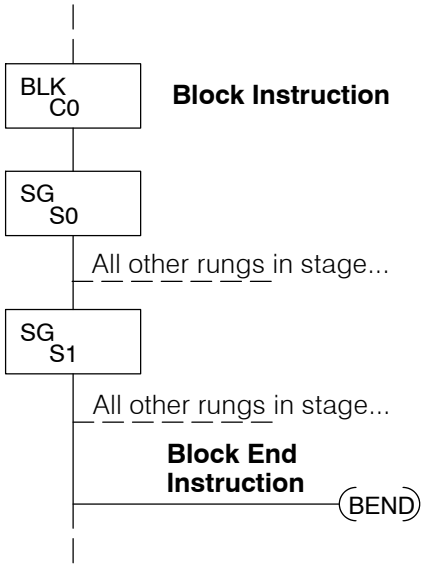
X	✓	✓
430	440	450

A block is a section of ladder program which contains stages. In the figure below, each block has its own reference number. Just like stages, a stage block may be active or inactive. Stages inside a block are not limited in how they may transition from one to another. Note that the use of stage blocks does not require each stage in a program to reside inside a block, shown below by the "stages outside blocks".



A program with 20 or more stages may be considered large enough to use block grouping (however, their use is not mandatory). When used, the number of stage blocks should probably be two or higher, because the use of one block provides a negligible advantage.

A block of stages is separated from other ladder logic with special beginning and ending instructions. In the figure to the right, the BLK instruction at the top marks the start of the stage block. At the bottom, the Block End (BEND) marks the end of the block. The stages in between these boundary markers (S0 and S1 in this case) and their associated rungs make up the block.



Note that the block instruction has a reference value field (set to "C0" in the example). The block instruction borrows or uses a control relay contact number, so that other parts of the program can control the block. *Any control relay number (such as C0) used in a BLK instruction is not available for use as a control relay.*

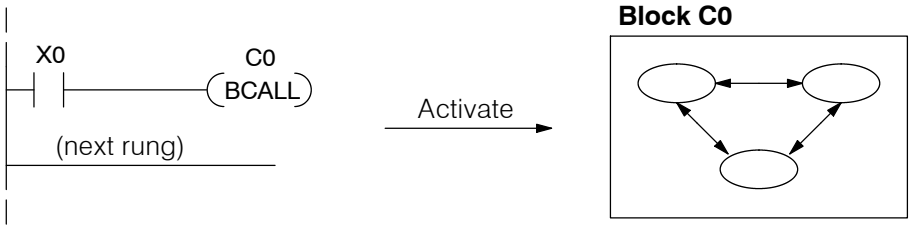
Note that the stages within a block must be regular stages (SG) or convergence stages (CV). So, they cannot be initial stages. The numbering of stages inside stage blocks can be in any order, and is completely independent from the numbering of the blocks.

Block Call (BCALL)

X	✓	✓
430	440	450

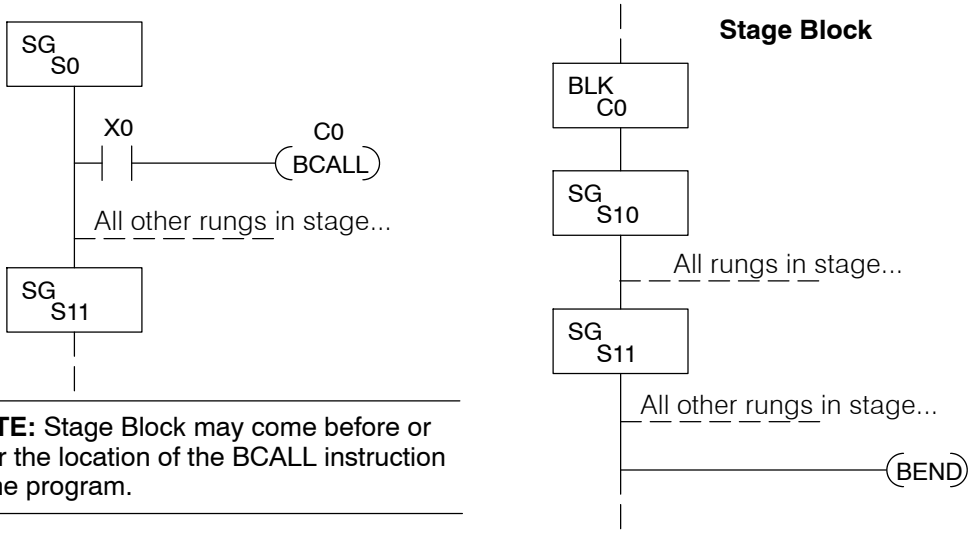
The purpose of the Block Call instruction is to activate a stage block. At powerup or upon Program-to-Run mode transitions, all stage blocks and the stages within them are inactive. Shown in the figure below, the Block Call instruction is a type of output coil. When the X0 contact is closed, the BCALL will cause the stage block referenced in the instruction (C0) to become active. When the BCALL is turned off, the corresponding stage block and the stages within it become inactive.

We must avoid confusing block call operation with how a “subroutine call” works. After a BCALL coil executes, program execution continues with the next program rung. Whenever program execution arrives at the ladder location of the stage block named in the BCALL, then logic within the block executes because the block is now active. Similarly, do not classify the BCALL as type of state transition (is not a JMP).



When a stage block becomes active, the first stage in the block automatically becomes active on the same scan. The “first” stage in a block is the one located immediately under the block (BLK) instruction in the ladder program. So, that stage plays a similar role to the initial type stage we discussed earlier.

The Block Call instruction may be used in several contexts. Obviously, the first execution of a BCALL must occur outside a stage block, since stage blocks are initially inactive. Still, the BCALL may occur on an ordinary ladder rung, or it may occur within an active stage as shown below. Note that either turning off the BCALL or turning off the stage containing the BCALL will deactivate the corresponding stage block. You may also control a stage block with a BCALL in another stage block.



NOTE: Stage Block may come before or after the location of the BCALL instruction in the program.

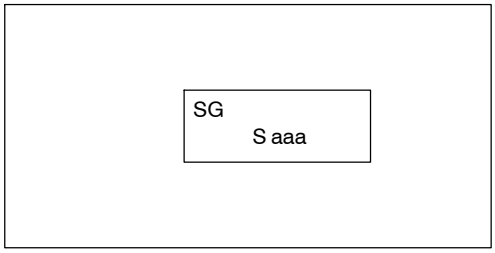
The BCALL may be used in many ways or contexts, so it can be difficult to find the best usage. Just remember that the purpose of stage blocks is to help you organize the application problem by grouping related stages together. Remember that initial stages must exist *outside* stage blocks.

RLL*PLUS* Instructions

Stage (SG)

430	440	450
✓	✓	✓

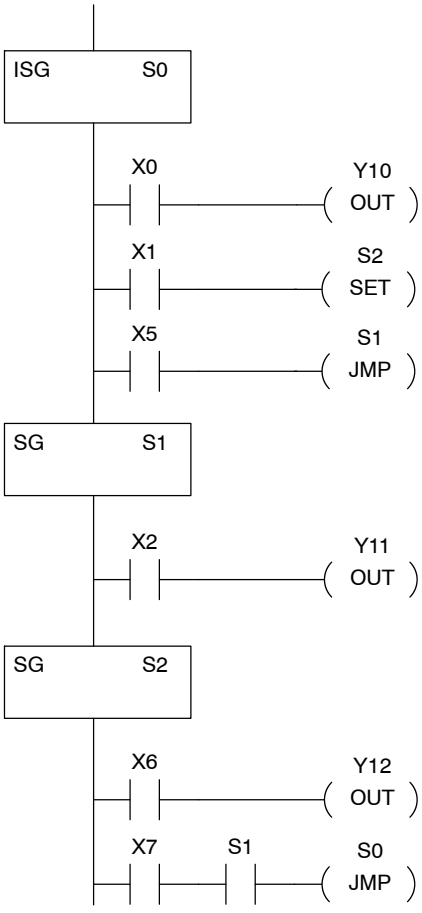
The Stage instructions are used to create structured RLL*PLUS* programs. Stages are program segments which can be activated by transitional logic, a jump or a set stage that is executed from an active stage. Stages are deactivated one scan after transitional logic, a jump, or a reset stage instruction is executed.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Stage	S	0-577	0-1777	0-1777

The following example is a simple RLL*PLUS* program. This program utilizes the initial stage, stage, and jump instruction to create a structured program.

DirectSOFT Display



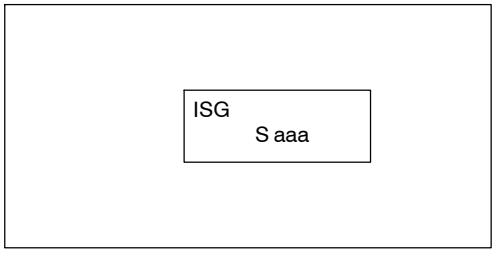
Handheld Programmer Keystrokes

ISG	S(SG)	0	↵	
STR	X(IN)	0	↵	
OUT	Y(OUT)	1	0	↵
STR	X(IN)	1	↵	
SET	S(SG)	2	↵	
STR	X(IN)	5	↵	
JMP	S(SG)	1	↵	
SG	S(SG)	1	↵	
STR	X(IN)	2	↵	
OUT	Y(OUT)	1	1	↵
SG	S(SG)	2	↵	
STR	X(IN)	6	↵	
OUT	Y(OUT)	1	2	↵
STR	X(IN)	7	↵	
AND	S(SG)	1	↵	
JMP	S(SG)	0	↵	

Initial Stage (ISG)

✓ ✓ ✓
430 440 450

The Initial Stage instruction is normally used as the first segment of an RLL*PLUS* program. Initial stages will be active when the CPU enters the run mode allowing for a starting point in the program. Initial Stages are also activated by transitional logic, a jump or a set stage executed from an active stage. Initial Stages are deactivated one scan after transitional logic, a jump, or a reset stage instruction is executed. Multiple Initial Stages are allowed in a program.



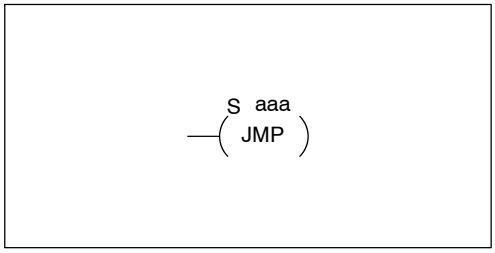
Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Stage	S	0-577	0-1777	0-1777

NOTE: If the ISG is within the retentive range for stages, the ISG will remain in the state it was in before power down and will NOT turn itself on during the first scan.

Jump (JMP)

✓ ✓ ✓
430 440 450

The Jump instruction allows the program to transition from an active stage which contains the jump instruction to another which stage is specified in the instruction. The jump will occur when the input logic is true. The active stage that contains the Jump will be deactivated 1 scan after the Jump instruction is executed.

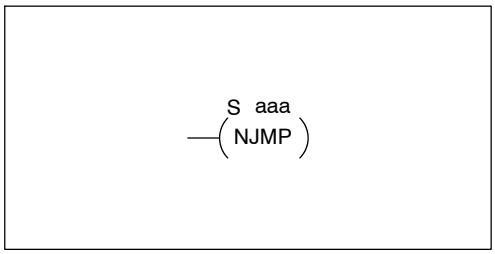


Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Stage	S	0-577	0-1777	0-1777

Not Jump (NJMP)

✓ ✓ ✓
430 440 450

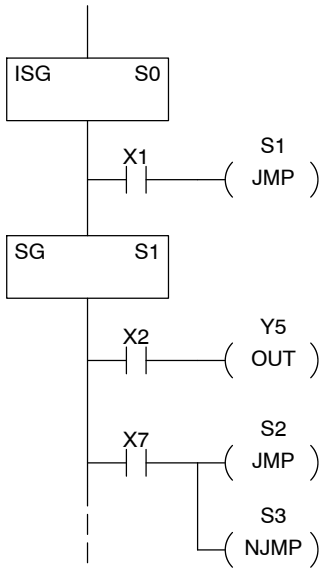
The Not Jump instruction allows the program to transition from an active stage which contains the jump instruction to another which is specified in the instruction. The jump will occur when the input logic is off. The active stage that contains the Not Jump will be deactivated 1 scan after the Not Jump instruction is executed.



Operand Data Type		DL430 Range	DL440 Range	DL450 Range
		aaa	aaa	aaa
Stage	S	0-577	0-1777	0-1777

In the following example, when the CPU begins program execution only ISG 0 will be active. When X1 is on, the program execution will jump from Initial Stage 0 to Stage 1. In Stage 1, if X2 is on, output Y5 will be turned on. If X7 is on, program execution will jump from Stage 1 to Stage 2. If X7 is off, program execution will jump from Stage 1 to Stage 3.

DirectSOFT Display



Handheld Programmer Keystrokes

ISG	S(SG)	0	↵
STR	X(IN)	1	↵
JMP	S(SG)	1	↵
SG	S(SG)	1	↵
STR	X(IN)	2	↵
OUT	Y(OUT)	5	↵
STR	X(IN)	7	↵
JMP	S(SG)	2	↵
NOT	JMP	S(SG)	3 ↵

Converge Stage (CV) and Converge Jump (CVJMP)

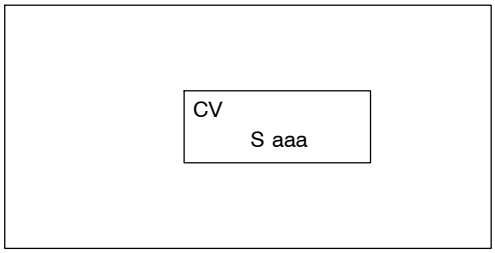
X	✓	✓
430	440	450

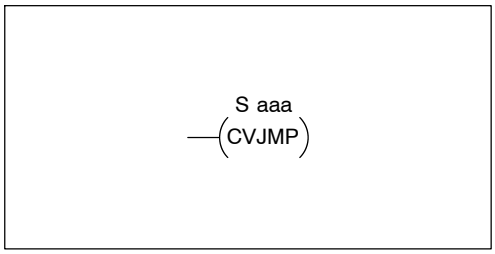
The Converge Stage instruction is used to group certain stages together by defining them as Converge Stages.

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 CVJUMP 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.

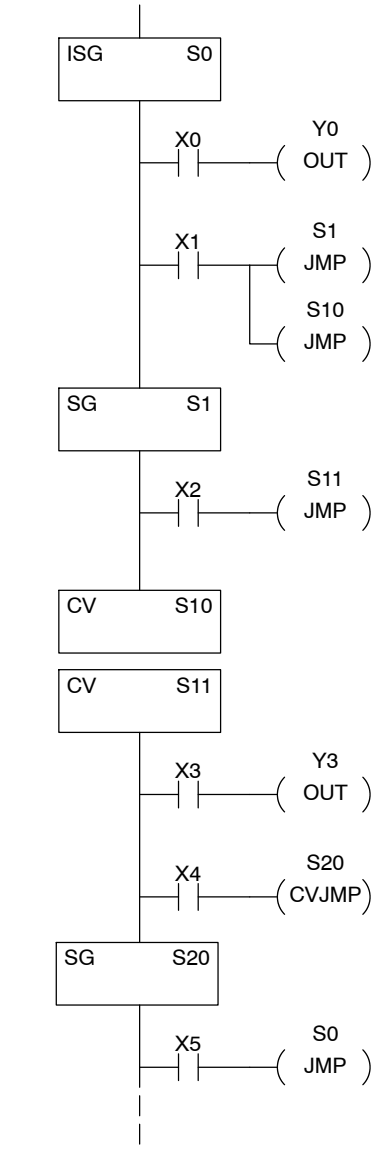




Operand Data Type		DL440 Range
		aaa
Stage	S	0–1777

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.

DirectSOFT Display



Handheld Programmer Keystrokes

ISG	S(SG)	0	↵				
STR	X(IN)	0	↵				
OUT	Y(OUT)	0	↵				
STR	X(IN)	1	↵				
JMP	S(SG)	1	↵				
JMP	S(SG)	1	0	↵			
SG	S(SG)	1	↵				
STR	X(IN)	2	↵				
JMP	S(SG)	1	1	↵			
SHFT	C	V	SHFT	S(SG)	1	0	↵
SHFT	C	V	SHFT	S(SG)	1	1	↵
STR	X(IN)	3	↵				
OUT	Y(OUT)	3	↵				
STR	X(IN)	4	↵				
SHFT	C	V	SHFT	JMP	S(SG)	2	0 ↵
SG	S(SG)	2	0	↵			
STR	X(IN)	5	↵				
JMP	S(SG)	0	↵				

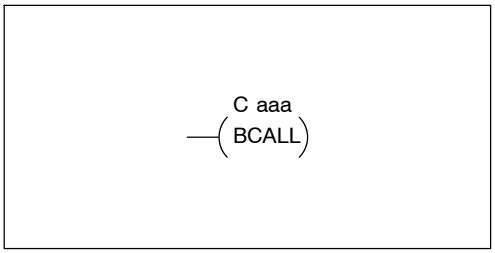
The stage block instructions are used to activate a block of stages. The Block Call, Block, and Block End instructions must be used together.

Block Call (BCALL)

X	✓	✓
430	440	450

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.*

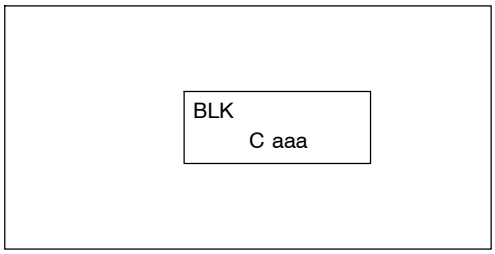
Activates First Block Stage — When the BCALL is executed it automatically activates the first stage following the BLK instructions.

Operand Data Type		DL440 Range
		aaa
Control Relay	C	0–1777

Block (BLK)

X	✓	✓
430	440	450

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.

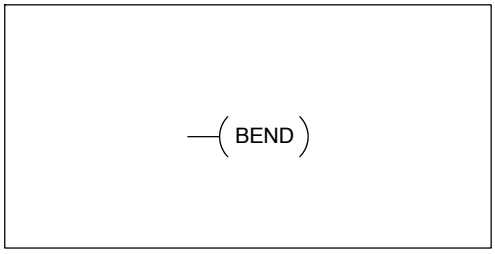


Operand Data Type		DL440 Range
		aaa
Control Relay	C	0–1777

Block End (BEND)

X	✓	✓
430	440	450

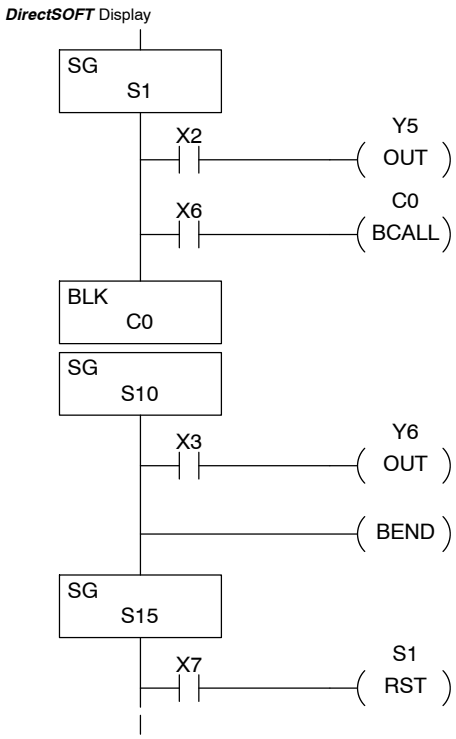
The Block End instruction is a label used with the Block instruction. It marks the end of a block of stages. There is no operand with this instruction.



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'll notice that X7 could reset Stage S1, which would disable the BCALL, thus resetting all stages within the block.

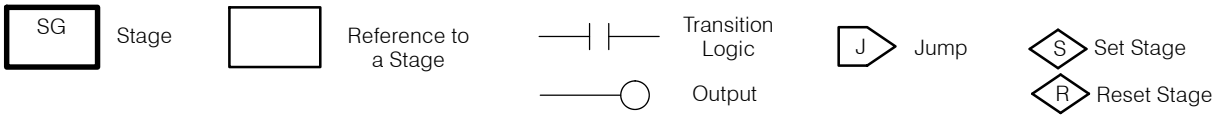


Handheld Programmer Keystrokes

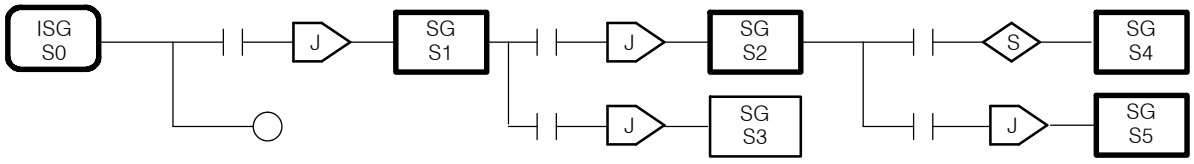
SG	S(SG)	1	↵						
STR	X(IN)	2	↵						
OUT	Y(OUT)	5	↵						
STR	X(IN)	6	↵						
SHFT	B	C	A	L	L	SHFT	C(CR)	0	↵
SHFT	B	L	K	SHFT	C(CR)	0	↵		
SG	S(SG)	1	0	↵					
STR	X(IN)	3	↵						
OUT	Y(OUT)	6	↵						
SHFT	B	E	N	D	↵				
SG	S(SG)	1	5	↵					
STR	X(IN)	7	↵						
RST	S(SG)	1	↵						

Stage View in *Direct*SOFT

The Stage View option in ***Direct***SOFT will let you view the ladder program as a flow chart. The figure below shows the symbol convention used in the diagrams. You may find the stage view useful as a tool to verify that your stage program has faithfully reproduced the logic of the state transition diagram you intend to realize.



The following diagram is a typical stage view of a ladder program containing stages. Note the left-to-right direction of the flow chart.



Questions and Answers about Stage Programming

We include the following commonly-asked questions about Stage Programming as an aid to new students. All question topics are covered in more detail in this chapter.

Q. What does stage programming do that I can't do with standard RLL programs?

A. Stages allow you to identify all the states of your process before you begin programming. This approach is more organized, because you divide a ladder program into sections. As stages, these program sections are active only when they are actually needed by the process. Most processes can be organized into a sequence of stages, connected by event-based transitions.

Q. Isn't a stage really just like a software subroutine?

A. No, it is very different. A subroutine is called by a main program when needed, and executes just once before returning to the point from which it was called. A stage, however, is part of the main program. It represents a state of the process, so an active stage executes on every scan of the CPU until it becomes inactive.

Q. What are Stage Bits?

A. A stage bit is 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 powerup.
- Another stage can execute a Stage JMP instruction naming this stage, which makes it active upon its next occurrence in the program.
- A program rung can execute a Set Stage Bit instruction (such as SET S0).

Q. How does a stage become inactive?

A. There are three ways:

- Standard Stages (SG) are automatically inactive at powerup.
- A stage can execute a Stage JMP instruction, resetting its Stage Bit to 0.
- Any rung in the program can execute a Reset Stage Bit instruction (such as RST S0).

Q. What about the power flow technique of stage transitions?

A. The power flow method of connecting adjacent stages (directly above or below in the program) actually is the same as the Stage Jump instruction executed in the stage above, naming the stage below. Power flow transitions are more difficult to edit in ***Direct***SOFT, we list them separately from two preceding questions.

Q. Can I have a stage 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 Stage Bit instruction when the current state is the last state in a sequence and its task is complete, or when a supervisory state is ending a state sequence under its command.

Q. What is an initial stage, and when do I use it?

A. An initial stage (ISG) is automatically active at powerup. Afterwards, it works just like any other stage. You can have multiple initial stages, if required. Use an initial stage for 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.

8

PID Loop Operation

(DL450 only)

In This Chapter. . . .

— DL450 PID Loop Features
— Introduction to PID Control
— Introducing DL450 PID Control
— PID Loop Operation
— Ten Steps to Successful Process Control
— PID Loop Setup
— PID Loop Tuning
— Using the Special PID Features
— Ramp/Soak Generator
— *Direct*SOFT Ramp/Soak Example
— Cascade Control
— Time Proportioning Control
— Feedforward Control
— PID Example Program
— Troubleshooting Tips
— Glossary of PID Loop Terminology
— Bibliography

DL450 PID Control

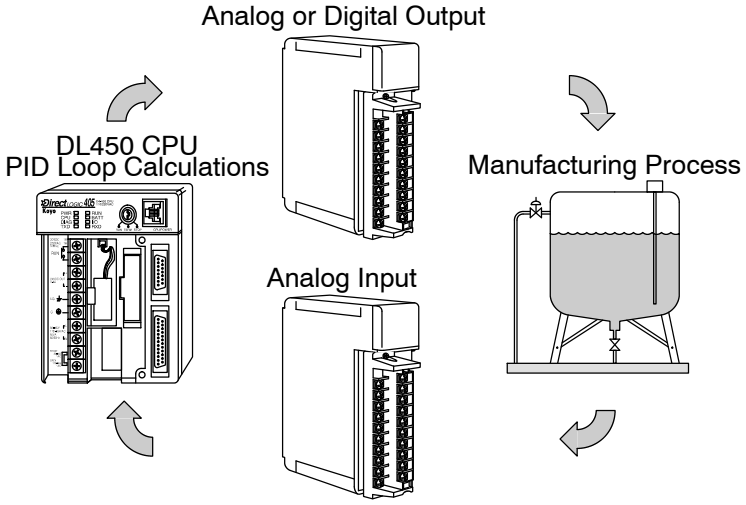
DL450 PID Control Features

Along with control functions discussed in this manual, the DL450 CPU features PID process control. The primary features are:

- Up to 16 loops, individual programmable sample rates
- The use of 16-bit analog inputs and outputs
- Manual/Automatic/Cascaded loop capability available
- Two types of bumpless transfer available
- Full-featured alarms
- Ramp/soak generator with up to 16 segments
- Auto Tuning

NOTE : The D4-450 CPU requires at least ***Direct***SOFT32, version 3.0c, build 58 or later and CPU firmware version 1.9 (HB) and 2.446 (SH) or later to implement these features. ***Direct***SOFT 5 is the latest programming software. See our website for more information **www.automationdirect.com**.

The DL450 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 and digital output modules. All process variables, gain values, alarm levels, etc., associated with each loop reside in a Loop Variable Table in the CPU. The DL450 CPU reads process variable (PV) inputs during each scan. Then it makes PID loop calculations during a dedicated time slice on each CPU scan, updating the control output value. The control loops use the Proportional-Integral-Derivative (PID) algorithm to generate the control output command. This chapter describes how the loops operate, and what you must do to configure and tune the loops.



As with ***Direct***SOFT32, ***Direct***SOFT 5 programming software, version 5.0 or later, is used for configuring analog control loops in the DL450. ***Direct***SOFT uses dialog boxes to help you set up the individual loops. After completing the setup, you can use ***Direct***SOFT's PID View to tune each loop. The configuration and tuning selections you make are stored in the DL450'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 , and 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 0.00 to 99.99 seconds
Rate Limits	Specify derivative gain limiting from 1 to 20
Bumpless Transfer I	Automatically initialized bias and setpoint when control switches from manual to automatic
Bumpless Transfer II	Automatically set 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?

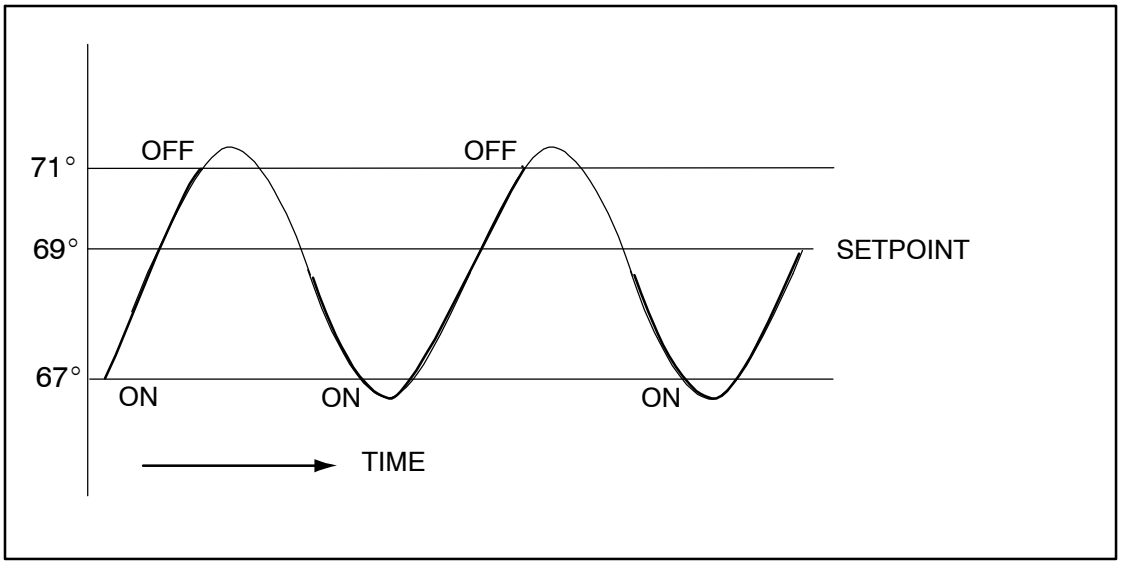
In this discussion, we will explain why PID control is used in process control instead of trying to provide control by simply using an analog input and a discrete output.

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 70 mph, the setpoint. Now, the car is cruising at a steady 70 mph until it comes to a hill to go up. As the car goes up the hill, it tends to slow down. The speed sensor senses this and causes the throttle to increase the fuel to the engine. The vehicle speeds up to maintain 70 mph without jerking the car and it reaches the top at the set speed. When the car levels out after reaching the top of the hill it will speed up. The speed sensor senses this and signals the throttle to provide less fuel to the engine, thus, the engine slows down allowing the car to maintain the 70 mph speed. How does this application apply to PID control? Lets look at the function of P, I and D terms:

- **Proportional** – is commonly referred to as Proportional Gain. The proportional term is the corrective action which is proportional to the error, that is, the change of the manipulated variable is equal to the proportional gain multiplied by the error (the activating signal). In mathematical terms:
  Proportional action = proportional gain X error
  Error = Setpoint (SP) – Process Variable (PV)
  Applying this to the cruise control, the speed was set at 70 mph which is the Setpoint. The speed sensor senses the actual speed of the car and sends this signal to the cruise controller as the Process Variable (PV). When the car is on a level highway, the speed is maintained at 70 mph, thus, no error since the error would be SP – PV = 0. When the car goes up the hill, the speed sensor detected a slow down of the car, SP–PV = error. The proportional gain would cause the output of the speed controller to bring the car back to the setpoint of 70 mph. This would be the Controlled Output.
- **Integral** – this term is often referred to as Reset action. It provides additional compensation to the control output, which causes a change in proportion to the value of the error over a period of time. In other words, the reset term is the integral sum of the error values over a period of time.
- **Derivative** – this term is referred to as rate. The Rate action adds compensation to the control output, which causes a change in proportion to the rate of change of error. Its job is to anticipate the probable growth of the error and generate a contribution to the output in advance.

Introducing DL450 PID Control

The DL450 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 DL450 CPU has the ability to directly accept signals from electronic sensors, such as thermocouples, pressure, VFDs, etc. These signals may be used in mathematically derived control systems.

In addition, the DL450 has built-in PID control algorithms that can be implemented. The basic function of PID closed loop process control is to maintain certain process characteristics at desired setpoints. As a rule, the process deviates from the desired setpoint reference as a result of load material changes and interaction with other processes. During this control, the actual condition of the process characteristics (liquid level, temperature, motor control, etc.) is measured as a process variable (PV) and compared with the target setpoint (SP). When deviations occur, an error is generated by the difference between the process variable (actual value) and the setpoint (desired value). Once an error is detected, the function of the control loop is to modify the control output in order to force the error to zero.

The DL450 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”

$$M(t) = Kc [e(t) + 1/Ti \int_0^t e(x)\, dx + Td\ d/dt\ e(t)] + M0$$

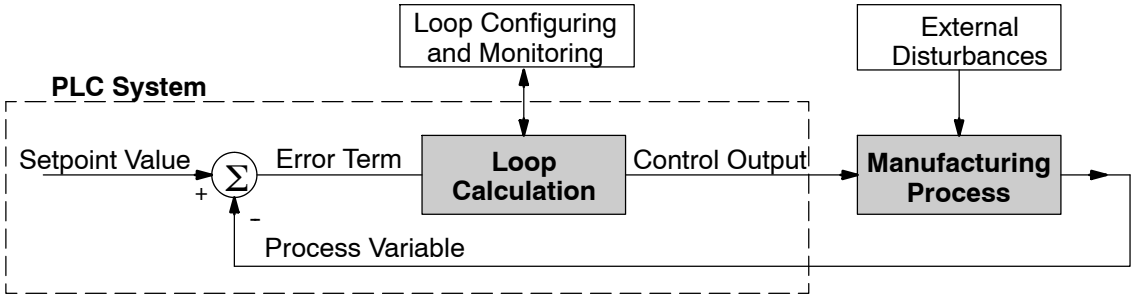
The analog input module receives the process variable in analog form along with an operator entered setpoint; the CPU computes the error. The error is used in the algorithm computation to provide corrective action at the control output. The function of the control action is based on an output control, 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 4–20mA 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 ***Direct***SOFT, additional ladder logic programming, both time proportioning (eg. heaters for temperature control) and position actuator (eg. 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 – a measurement of some physical property of the raw materials. Measurements are made using some type of sensor. For example, if the manufacturing process uses an oven, we will have a strong interest in controlling temperature. Therefore, temperature is a process variable.

Setpoint Value – the theoretically perfect quantity of the process variable, or the desired amount which yields the best product. The machine operator knows this value, and either sets it manually or programs it into the PLC for later automated use.

External Disturbances – the unpredictable sources of error which the control system attempts to cancel by offsetting their effects. For example, if the fuel input is constant an oven will run hotter during warm weather than it does during cold weather. An oven control system must counter-act this effect to maintain a constant oven temperature during any season. Thus, the weather (which is not very predictable), is one source of disturbance to this process.

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.

Loop Calculation – the real-time application of a mathematical algorithm to the error term, generating a control output command appropriate for minimizing the error magnitude. Various control algorithms are available, and the DL450 uses the Proportional-Derivative-Integral (PID) algorithm (more on this later).

Control Output – the result of the loop calculation, which becomes a command for the process (such as the heater level in an oven).

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 DL450 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 DL450 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.

PID Position Algorithm

Referring to the control output equation on page 8-6, the DL450 CPU approximates the output M(t) using a discrete position form of the PID equation.

Let:

Ts = Sample rate
Kc = Proportional gain
Ki = Kc * (Ts/Ti) = Coefficient of integral term
Kr = Kc * (Td/Ts) = Coefficient of derivative term
Ti = Reset or integral time
Td = Derivative time or rate
SP = Setpoint
PV_n = Process variable at n^{th} sample
en = SP - PV_n = Error at n^{th} sample
Mo = Value to which the controller output has been initialized

Then:

M_n = Control output at n^{th} sample

$$M_n = K_c * e_n + K_i \sum_{i=1}^{n} e_i + K_r (e_n - e_{n-1}) + M_o$$

This form of the PID equation is referred to as the position form since the actual actuator position is computed. The velocity form of the PID equation computes the change in actuator position. The CPU modifies the standard equation slightly to use the derivative of the process variable instead of the error as follows:

$$M_n = K_c * e_n + K_i \sum_{i=1}^{n} e_i + K_r (PV_n - PV_{n-1}) + M_o$$

These two forms are equivalent unless the setpoint is changed. In the original equation, a large step change in the setpoint will cause a correspondingly large change in the error resulting in a bump to the process due to derivative action. This bump is not present in the second form of the equation.

The DL450 also combines the integral sum and the initial output into a single term called the bias (Mx). This results in the following set of equations:

$Mx_o = M_o$

$Mx = Ki * e_n + Mx_{n-1}$

$M_n = Kc * e_n - Kr(PV_n - PV_{n-1}) + Mx_n$

The DL450 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 DL450 also allows you to specify the minimum and maximum output limit values (within the range 0 to 4095 in binary 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:

$Mx = Ki * e_n + Mx_{n-1}$

For example, assume the output is controlling a valve and the PV remains at some value greater than the setpoint. The negative error (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 DL450 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 DL450 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.

$Mx = Ki * e_n + Mx_{n-1}$

$M = Kc * e_n - Kr(PV_n - PV_{n-1}) + Mx$

$M_n = 0$ "if $M < 0$"
$M_n = M$ "if $0 < M > 1$"
$M_n = 1$ "if $M > 1$"

$Mx_n = Mx$ "if $0 < M > 1$"
$Mx_n = Mx_{n-1}$ "otherwise"

Thus in this example, the bias will probably not go all the way to zero so that, when the PV does begin to come down, the loop will begin to open the valve sooner than it would have if the bias had been allowed to go all the way to zero. This action has the effect of reducing the amount of overshoot.

Adjusting the Bias

The normal action of the CPU is to adjust the bias term when the output goes out of range as shown below.

$Mx = Ki * e_n + Mx_{n-1}$

$M = Kc * e_n - Kr(PV_n - PV_{n-1}) + Mx$

$M_n = 0$ "if $M < 0$"
$M_n = M$ "if $0 < M > 1$"
$M_n = 1$ "if $M > 1$"

$Mx_n = Mx$ "if $0 < M > 1$"
$Mx_n = M_n - Kc * e_n - Kr(PV_n - PV_{n-1})$ "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 SP

This feature reduces oscillation caused by a step change in setpoint when the adjusting bias feature is used.

$Mx = Mx * SP_n / SP_{n-1}$ if the loop is direct acting

$Mx = Mx * SP_{n-1} / SP_n$ if the loop is reverse acting

$Mx_n = 0$ "if $Mx < 0$"
$Mx_n = Mx$ "if $0 < Mx > 1$"
$Mx_n = 1$ "if $M > 1$"

Eliminating Proportional, Integral or Derivative Action

It is not always necessary to run a full three mode PID control loop. Most loops require only the PI terms or just the P term. Parts of the PID equation may be eliminated by choosing appropriate values for the gain (Kc), reset (Ti) and rate (Td) yielding a P, PI, PD, I and even an ID and a D loop.

Eliminating Integral Action

The effect of integral action on the output may be eliminated by setting Ti = 9999 or 0000. 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:

$$\triangle M_n = M - M_{n-1}$$

Bumpless Transfer

The DL450 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 = Control Ouput	

The bumpless transfer feature of the DL450 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 DL450 allows the user to specify alarm conditions that are to be monitored for each loop. Alarm conditions are reported to the CPU by setting up the alarms in ***Direct***SOFT using the PID setup alarm dialog when the loop is setup. The alarm features for each loop are:

- **PV Limit** — Specify up to four PV alarm points.
  - **High-High** PV rises above the programmed High-High Alarm Limit.
  - **High** PV rises above the programmed High Alarm Limit.
  - **Low** PV fails below the Low Alarm Limit.
  - **Low-Low** PV fails below the Low-Low Limit.
- **PV Deviation Alarm** — Specify an alarm for High and Low PV deviation from the setpoint (Yellow Deviation). An alarm for High High and Low Low PV deviation from the setpoint (Orange Deviation) may also be specified. When the PV is further from the setpoint than the programmed Yellow or Orange Deviation Limit the corresponding alarm bit is activated.
- **Rate-of-Change** — This alarm is set when the PV changes faster than a specified rate-of-change limit.
- **PV Alarm Hysteresis** — The PV Limit Alarms and PV Deviation Alarms are programmed using threshold values. When the absolute value or deviation exceeds the threshold, the alarm status becomes true. Real-world PV signals have some noise on them, which can cause some fluctuation in the PV value in the CPU. As the PV value crosses an alarm threshold, its fluctuations will cause the alarm to be intermittent and annoy process operators. The solution is to use the PV Alarm Hysteresis feature.

Loop Operating Modes

The DL450 loop controller operates in one of two modes, either Manual or Automatic.

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 DL450 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 DL450 loop controller allows a loop to operate as reverse acting. With a reverse acting loop, the output is driven in the opposite direction of the error. For example, if SP > PV, then a reverse acting controller will decrease the output to increase the PV.

$$Mx = -Ki * e_n + Mx_{n-1}$$

$$M = -Kc * e_n + Kr(PV_n - PV_{n-1}) + Mx_n$$

Square Root of the Process Variable

Square root is selected whenever the PV is from a device such as an orifice meter which requires this calculation.

Error Squared Control

Whenever error squared control is selected, the error is calculated as:

$$e_n = (SP - PV_n) * ABS(SP - PV_n)$$

A loop using the error squared is less responsive than a loop using just the error, however, it will respond faster with a large error. The smaller the error, the less responsive the loop. Error squared control would typically be used in a pH control application.

Error Deadband Control

With error deadband control, no control action is taken if the PV is within the specified deadband area around the setpoint. The error deadband is the same above and below the setpoint.

Once the PV is outside of the error deadband around the setpoint, the entire error is used in the loop calculation.

$e_n = 0$ "SP – Deadband__Below_SP < PV < SP – Deadband_Above_SP"
$e_n = P - PV_n$ "otherwise"

The error will be squared first if both Error Squared and Error Deadband 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, Y_n, as shown below.

$$Y_n = Y_{n-1} + \frac{Ts}{Ts + \left(\frac{Td}{Kd}\right)} * (PV_n = Y_{n-1})$$

Position Algorithm

$$Mx = Ki * e_n + Mx_{n-1}$$
$$M = Kc * e_n - Kr * (Y_n - Y_{n-1}) + Mx$$

Velocity Algorithm

$$\Delta M = Kc * (e_n - e_{n-1}) + Ki * e_n - Kr * (Y_n - 2 * Y_{n-1} + Y_{n-2})$$

Ten Steps to Successful Process Control

Modern electronic controllers such as the DL450 CPU provide sophisticated process control features. Automated control systems can be very 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 knowledge 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 of the 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

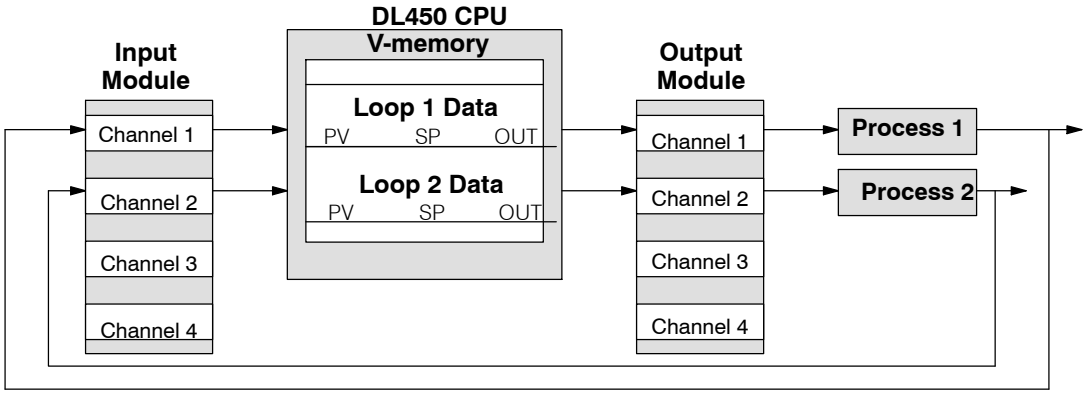
Assuming the control strategy is sound, it is still crucial to *properly size the actuators and properly scale the sensors*.

- 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 DL450 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 modules. Refer to the figure on the next page. In many cases, you will be able to share input or output modules among several control loops. The example shown sends the PV and Control Output signals for two loops through the same set of modules. Up to four loops could be handled by the modules shown.

AutomationDirect offers DL405 analog modules with 4, 8, and 16 channels per module in various signal types and ranges. Also available are thermocouple and RTD modules which can be used to maintain temperatures to within a 10^{th} of a degree. Refer to our sales catalog for further information on these modules, or find the modules on our website, **www.automationdirect.com**.



Step 5: Wiring and Installation

After selection and procurement of all loop components and I/O modules, we can perform the wiring and installation. Refer to the wiring guidelines in Chapter 2 of this Manual, and to the DL405 Analog I/O Module manual as needed. The most commonly overlooked wiring details 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 by using ***Direct***SOFT. 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 completed, and loop parameters entered, the Manual mode must be used to manually and carefully check out the new control system.

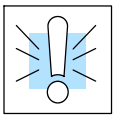
- Verify that the PV value from the sensor is correct.
- If it is safe to do so, gradually increase the control output up above 0%, and see if the PV responds (*and moves in the correct direction!*).

Step 8: Loop Tuning

If the open loop test (page 8-40) shows the PV reading is good and the control output has the proper effect on the process; follow the closed-loop auto tuning procedure (see page 8-45). In this step, the loop is tuned so the PV automatically follows the SP.

Step 9: Run Process Cycle

If the closed loop test shows PV will follow small changes in the SP, consider running an actual process cycle. The programming which will generate the desired SP in real time must be completed. In this step, it may desirable to run a small test batch of product through the machine, while 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 Loop 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 DL405 Analog I/O Modules Manual, D4-ANLG-M). The DL450 CPU 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	V1400 - V7340, V10000 - V37740	write
V7641	Number of Loops	BCD	0 - 16	write
V7642	Loop Error Flags	Binary	0 or 1	read

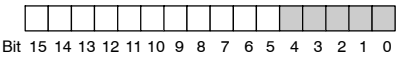
If the number of loops is "0", the loop controller task is turned off during the ladder program scan. The loop controller will allow use of loops in ascending order, beginning with 1. For example, you cannot use loop 1 and 4 while skipping 2 and 3. The loop controller attempts to control the full number of loops specified in V7641.

NOTE: The V-memory data is stored in RAM memory. If power is removed from the CPU for an extended period of time, the PID Setup Parameters will be lost. It is recommended to use the optional D3-D4-BAT for memory backup.

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.

PID Error Flags, V7642



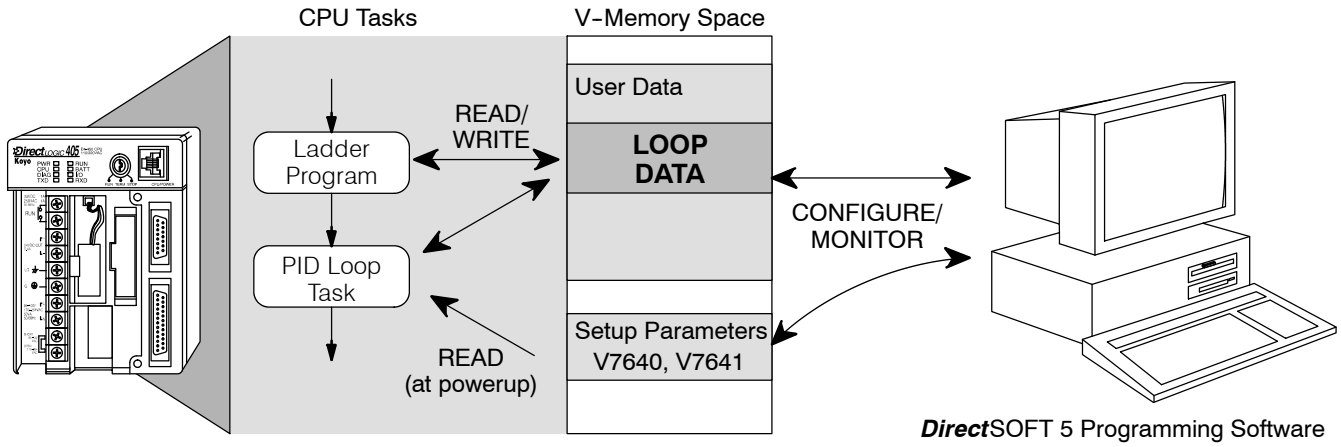
If you use the ***Direct***SOFT loop setup dialog box, its automatic range checking prohibits possible setup errors. However, the setup parameters may be written using other methods such as RLL, so the error flag register may be helpful in those cases. The following table lists the errors reported in V7642.

Bit	Error Description (0 = no error, 1 = error)
0	The starting address (in V7640) is out of the lower V-memory range.
1	The starting address (in V7640) is out of the upper V-memory range.
2	The number of loops selected (in V7641) is greater than 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 V37777. Use an address closer to V10000.

As a quick check, if the CPU is in Run mode and V7642=0000, then we know there are no programming errors.

Establishing the Loop Table Size and Location

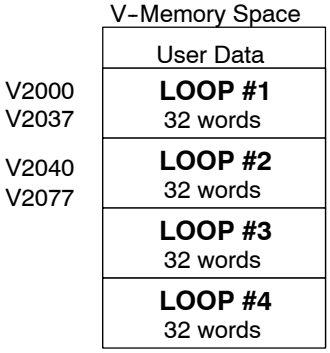
On a PROGRAM-to-RUN mode transition, the CPU reads the loop setup parameters as pictured below. At that moment, the CPU learns the location of the loop table and the number of loops it configures. Then during the ladder program scan, the PID Loop task uses the loop data to perform calculations, generate alarms, and so on. There are some loop table parameters the CPU will read or write on every loop calculation.



NOTE: The DL450 CPU's PID algorithm requires at least ***Direct***SOFT32, version 3.0c, Build 58 (or later) and CPU firmware version 1.9 (H8) and 2.446 (SH) (or later), or ***Direct***SOFT 5, version 5.0 (or later). See our website for more information: **www.automationdirect.com**.

The Loop Table contains data for only the number of loops 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.

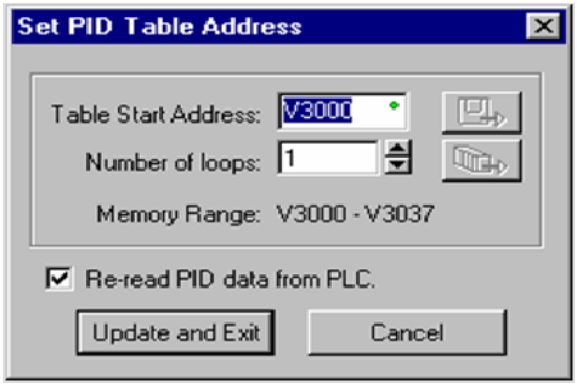


Determine the block of V-memory to be used for each PID loop. Besides being the beginning of the PID parameter memory block, the first address will be the start of loop 1 parameters. Remember, there are 32 words (0 to 37 octal) needed for each loop. Once you have determined the beginning V-memory address to be used, you can setup and store the PID parameters either directly in your RLL program or by the using PID Setup in ***Direct***SOFT.

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 ***Direct***SOFT is the simplest way to setup the parameters. The DL450 PID parameters can be setup either offline or online while developing the user program. The parameters will be loaded to V-memory as the program is loaded into the PLC. If the PID parameters are setup or changed while the PLC is connected to the programming computer, this can only be done in Program Mode.

To begin the PID setup, open an edited program with ***Direct***SOFT, then click on **PLC > Setup > PID** to access the Setup PID dialog which is pictured below.



First type the beginning address in the PID Table Address dialog. 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.

Loop Table Word Definitions

The parameters associated with each loop are listed in the following table. The address offset is in octal, to help you locate specific parameters in a 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# 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	reserved for future use	-	-
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	reserved for future use	-	-
32	Addr + 37	reserved for future use	-	-

* Read data only when alarm enable bit transitions 0 to1, ** 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 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	reserved for future use	–	–	–

PID Mode Setting 2 Descriptions (Addr + 01)

The bit definitions for PID Mode Setting 2 word (Addr+01) are listed in the following table. More information about the use of this word is available later in this chapter.

Bit	PID Mode Setting 2 Description	Read/Write	Bit=0	Bit=1
0	Input (PV) and Control Output Range Unipolar/Bipolar select (See Notes 1 and 2)	write	unipolar	bipolar
1	Input/Output Data Format select (See Notes 1 and 2)	write	12 bit	15 bit
2	reserved for future use	-	-	-
3	SP Input limit enable	write	disable	enable
4	Integral Gain (Reset) units select	write	seconds	minutes
5	Select Autotune PID algorithm	write	closed loop	open loop
6	Autotune selection	write	PID	PI only (rate = 0)
7	Autotune start (Note: Bit 7 can be used to cancel Autotune mode by setting it to 0.)	read/write	autotune done	force start
8	PID Scan Clock (internal use)	read	-	-
9	Input/Output Data Format 16-bit select (See Notes 1 and 2)	write	not 16 bit	select 16 bit
10	Select separate data format for input and output (See Notes 2 and 3)	write	same format	separate formats
11	Control Output Range Unipolar/Bipolar select (See Notes 2 and 3)	write	unipolar	bipolar
12	Output Data Format select (See Notes 2 and 3)	write	12 bit	15 bit
13	Output data format 16-bit select (See Notes 2 and 3)	write	not 16 bit	select 16 bit
14-15	Reserved for future use	-	-	-

NOTE 1: 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 2: 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 3: 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 word (Addr+06) 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
12-15	Reserved for Future Use	-	-	-

Ramp/Soak Table Flags (Addr + 33)

The individual bit definitions of the Ramp/Soak Table Flag word (Addr+33) 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→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)	

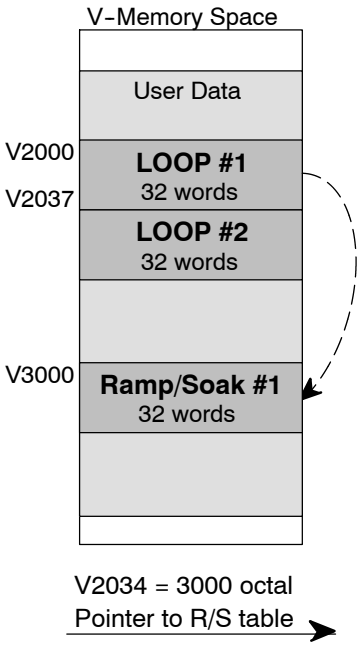
Bits 8-15 must be read as a byte to indicate the current segment number of the Ramp/Soak generator in the profile. This byte will have the values 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, and 10. which represent segments 1 to 16 respectively. If the byte=0. then the Ramp/Soak table is not active.

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 in a continuous stream, called a profile. To use the Ramp/Soak feature, you must program a separate table of 32 words with appropriate values. A ***Direct***SOFT dialog box makes this easy to do.

In the basic 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 be adjoining 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 Generator section 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

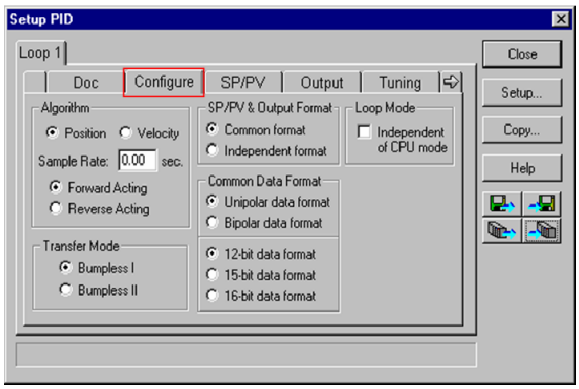
Ramp/Soak Table Programming Error Flags (Addr + 35)

The individual bit definitions of the Ramp/Soak Table Programming Error Flags (Addr + 35) word are listed in the following table.

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 out of 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 ***Direct***SOFT 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.



Select the Algorithm Type

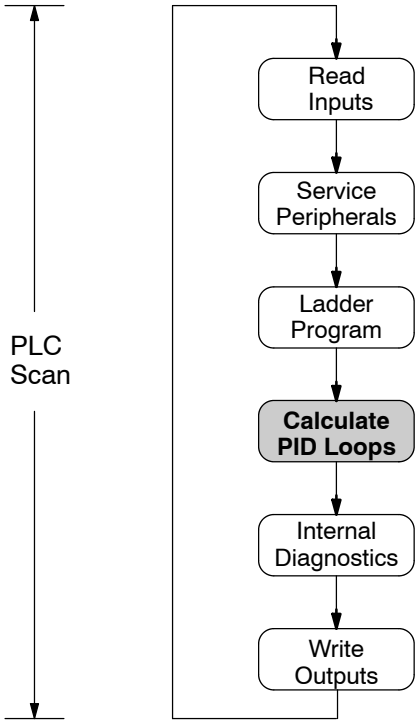
Chose either *Position or Velocity*. The default algorithm is Position. This is the choice for most applications which include heating and cooling loops as well as most position and level control loops. A typical velocity control will consist of a process variable such as a flow totalizer in a flow control 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 DL450 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 calculating 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 output 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).

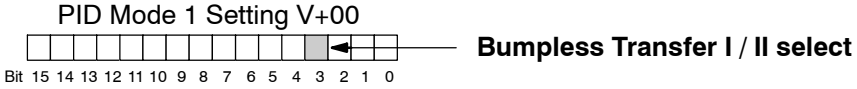
The Transfer Mode

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	PID Algorithm	Manual-to-Auto Transfer Action	Auto-to-Cascade Transfer Action
Bumpless Transfer I	0	Position	Forces Bias = Control Output Forces SP = PV	Forces Major Loop Output = Minor Loop PV
		Velocity	Forces SP = PV	Forces Major Loop Output = Minor Loop PV
Bumpless Transfer II	1	Position	Forces Bias = Control Output	none
		Velocity	none	none

The transfer type can also be selected in a RLL program by setting bit 3 of PID Mode 1, V+00 setting as shown.

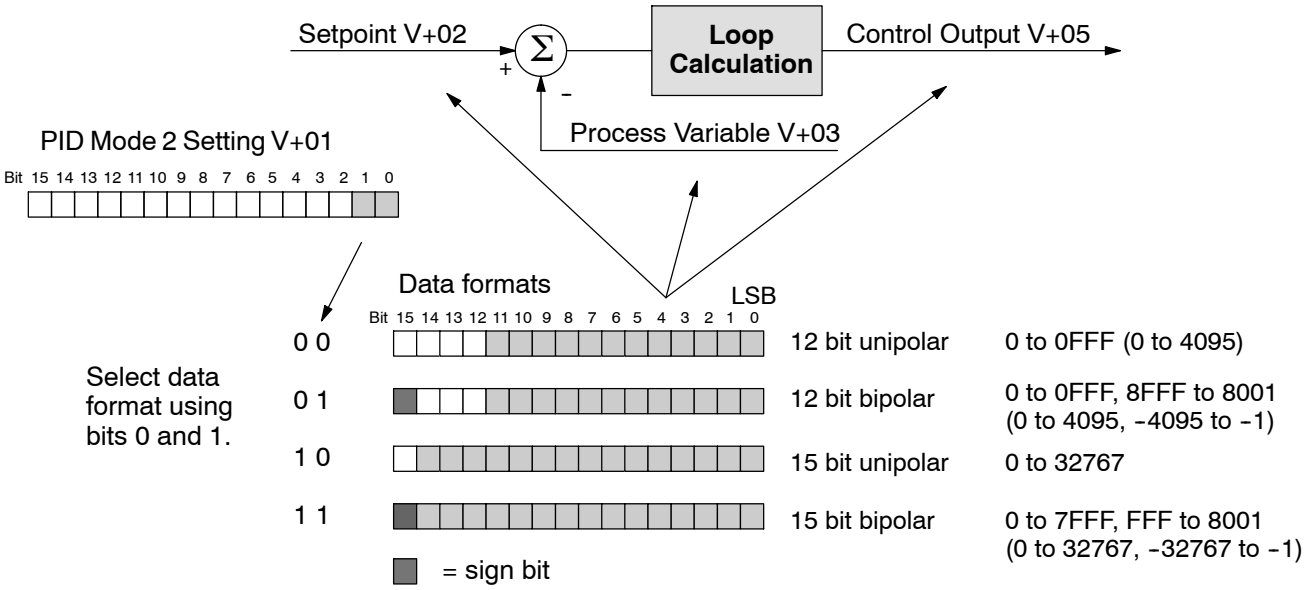


SP/PV & Output Format

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.

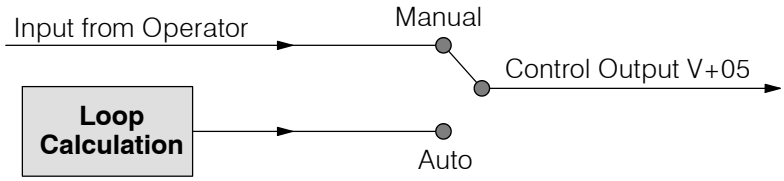


The data format determines the numerical interface between the PID loop and the PV sensor and the control output device. This selects the data format for both the SP and the PV.

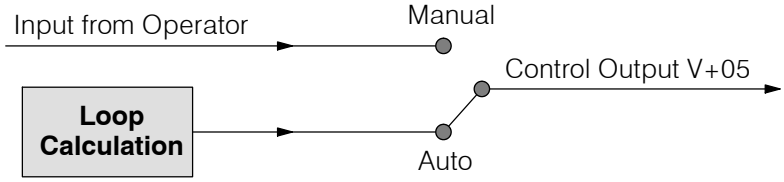
Loop Mode

The feature called *Independent of CPU mode* in the dialog is not available in the DL450. However, the DL450 does provide 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.

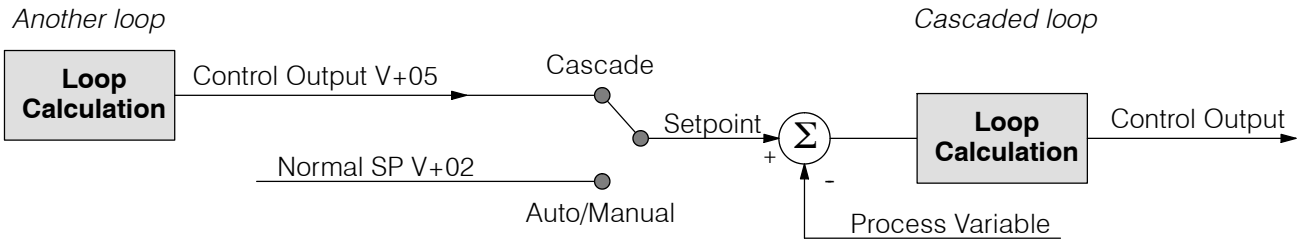
In Manual Mode, the loop is not executing PID calculations (however, loop alarms are still active). With regard to the loop table, the CPU stops writing values to location V+05 (control output) for that loop. *It is expected that an operator or other intelligent source* is manually controlling the output 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 just as in Automatic Mode, with one important change. The data source for the SP changes from its normal location at V+02, using the control output value from another loop (the purpose of cascading loops is covered later in this chapter). 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.

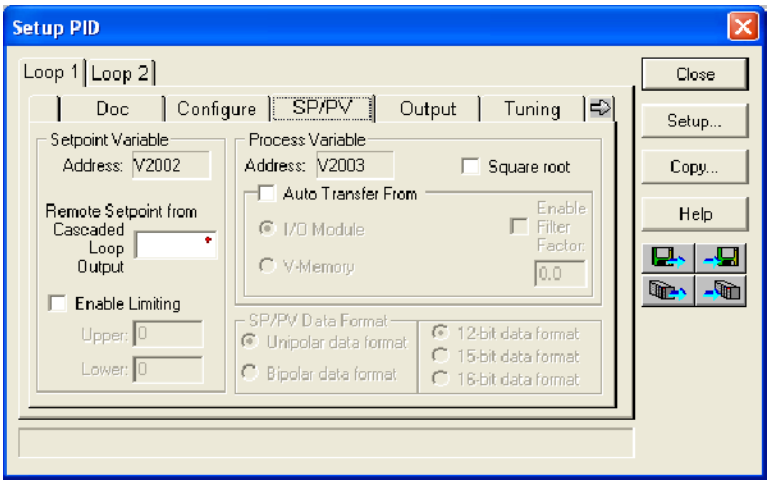


As pictured below, a loop can change from one mode to another, but *cannot go from Manual Mode to Cascade*. This mode change is prohibited because a loop would be changing two data sources at the same time, and could cause a loss of control.

When 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.

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. The *Auto Transfer From I/O module* will be grayed out and not available for use by the DL450.



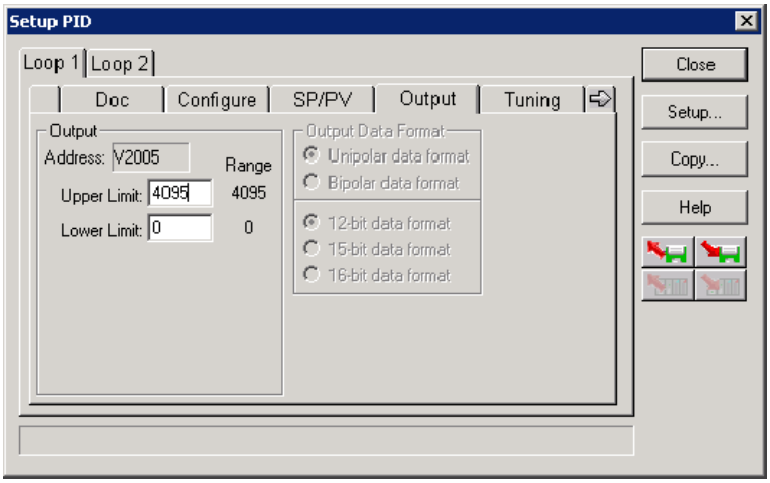
NOTE: Auto Transfer From I/O module **is not available** in the DL450 PLC at this time, but will be available in the future.

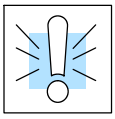
If the Auto Transfer From I/O module is selected, an error message (shown below) will appear.

NOTE: The SP/PV dialog can be left as it first appears for basic PID operations.

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. The *Auto transfer to I/O module* is not available for use by the DL450.The *Output Data format* area is not available and is grayed out if Common format has been chosen (see page 8-26).

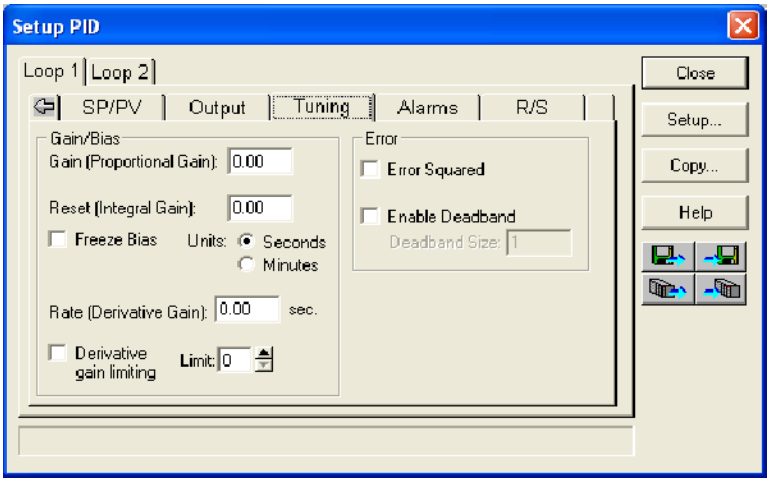




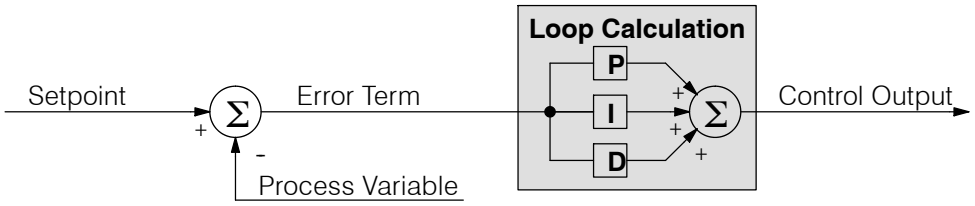
WARNING: If the Upper Limit is set to zero, the output will never get above zero. In effect, there will be no control output.

Enter PID Tuning Parameters

Another PID setup dialog, Tuning, is for entering the PID parameters shown as: Gain (Proportional Gain), Reset (Integral Gain) and Rate (Derivative Gain)



Recall the position and velocity forms of the PID loop equations which were introduced earlier. The equations basically show the three components of the PID calculation: Proportional Gain (P), Integral Gain (I) and Derivative Gain (D). The following diagram shows a form of the PID calculation in which the control output is the sum of the proportional gain, integral gain and derivative gain. With each calculation of the loop, each term receives the same error signal value.



The P, I and D gains are 4-digit BCD numbers with values from 0000 to 9999. They contain an implied decimal point in the middle, so the values are actually 00.00 to 99.99. Some gain values have units — Proportional gain has no unit, Integral gain may be selected in seconds or in minutes, and Derivative gain is in seconds.

Gain (Proportional Gain) — This is the most basic gain of the three. Values range from 0000 to 9999, but they are used internally as xx.xx. An entry of "0000" effectively removes the proportional term from the PID equation. This accommodates applications which need integral-only loops.

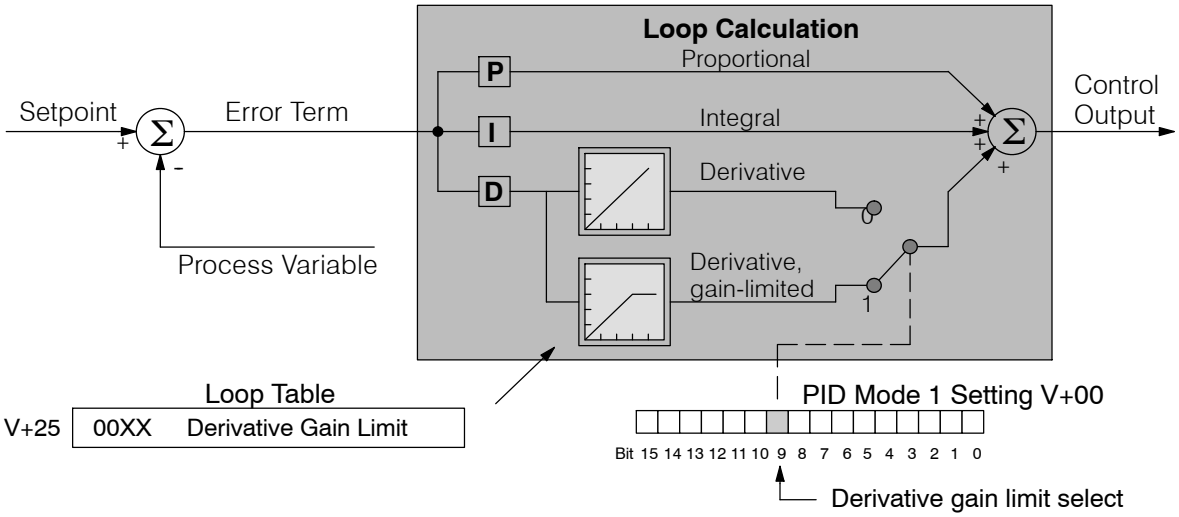
Reset (Integral Gain) — Values range from 0001 to 9998, but they are used internally as xx.xx. An entry of "0000" or "9999" causes the integral gain to be "∞", effectively removing the integrator term from the PID equation. This accommodates applications which need proportional-only loops. The units of integral gain may be either seconds or minutes, as shown in the above dialog.

Rate (Derivative Gain) — Values which can be entered range from 0001 to 9999, but they are used internally as xx.xx. An entry of "0000" allows removal of the derivative term from the PID equation (a common practice). This accommodates applications 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 ***Direct***SOFT 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 Derivative Gain Limiting is selected, a unit of 0–20 for ***Limit*** must also be entered.

NOTE: When first configuring a loop, it's best to use the standard error term until after the loop is tuned. Once the loop is tuned, you will be able to tell if these functions will enhance control. The Error Squared and/or Enable Deadband can be selected later in the PID setup. Also, values are not required to be entered in the Tuning dialog, but they can set later in the ***Direct***SOFT PID View.

Error Term Selection

The error term is internal to the CPUs PID loop controller, and is generated again in each PID calculation. Although its data is not directly accessible, you can easily calculate it by subtracting: Error = (SP – PV). The PID calculation operates on this value linearly to give the result. However, a few applications can benefit from non–linear control. The Error–squared method of non–linear control exaggerates large errors and diminishes small error

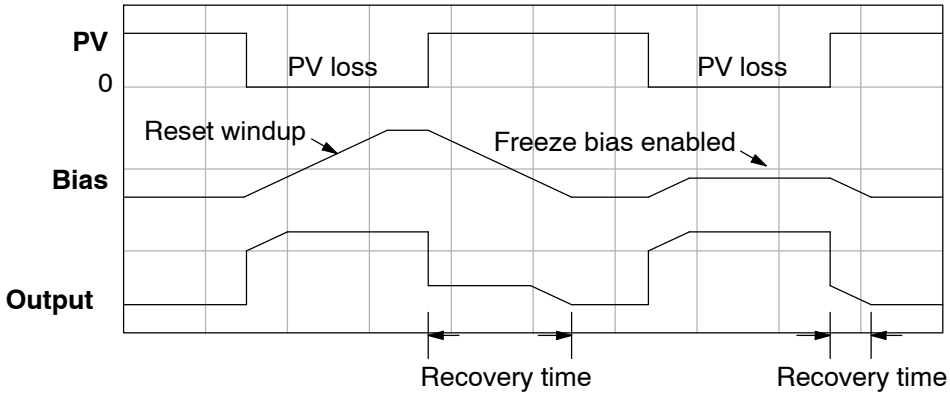
Error Squared — When selected, the squared error function simply squares the error term (but preserves the original algebraic sign), which is used in the calculation. This affects the Control Output by diminishing its response to smaller error values, but maintaining its response to larger errors. Some 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 to its range limits. Much of the reset windup is thus avoided, and the output recovery time is much less.

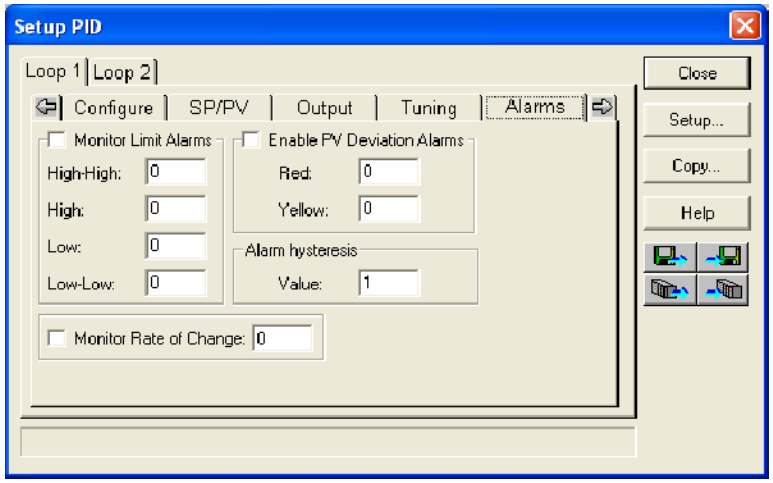
For most applications, the freeze bias feature will work with the loop as described above. It is suggested to enable this feature by selecting it in the dialog. Bit 10 of PID Mode 1 Setting (V+00) word can also be set in RLL.

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

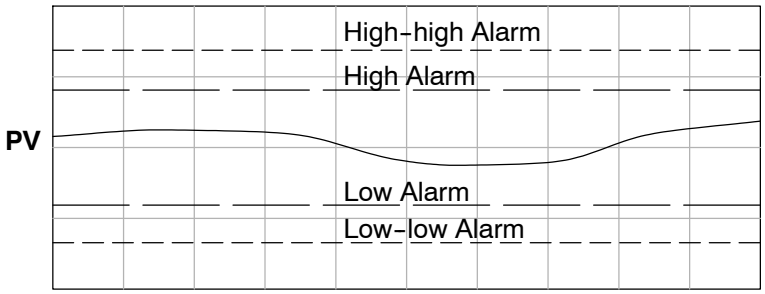
Although the setup of the PID alarms is optional, you surely would not want to operate a process without monitoring it. The performance of a process control loop may generally be measured by how closely the process variable matches the setpoint. Most process control loops in industry operate continuously, 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.



Monitor Limit Alarms

Checking this box will allow all of the PV limit alarms to be monitored once the limits are entered. The PV absolute value alarms are organized as two upper and two lower alarms. The alarm status is false as long as the PV value remains in the region between the upper and lower alarms, as shown below. The alarms nearest the safe zone are named *High Alarm* and *Low Alarm*. If the loop loses control, the PV will cross one of these thresholds first. Therefore, you can program the appropriate alarm threshold values in the loop table locations shown below to the right. The data format is the same as the PV and SP (12-bit or 15-bit). The threshold values for these alarms should be set to give an operator an early warning if the process loses control.



Loop Table

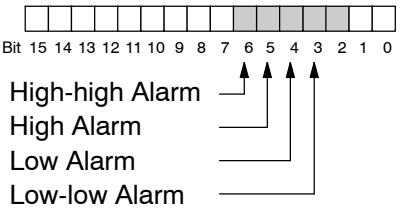
V+16	XXXX	High-high Alarm
V+15	XXXX	High Alarm
V+14	XXXX	Low Alarm
V+13	XXXX	Low-low Alarm

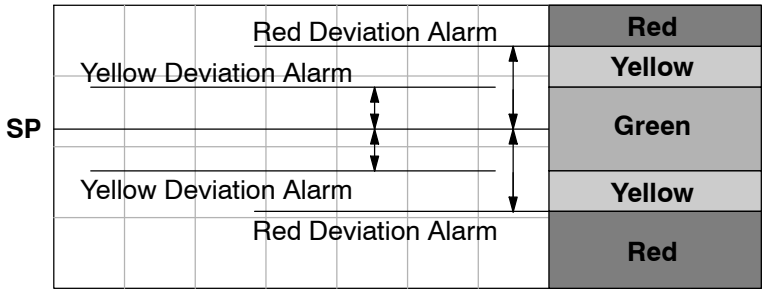
NOTE: The Alarm dialog can be left as it first appears, without alarm entries. The alarms can then be setup in the ***Direct***SOFT PID View.

If the process remains out of control for some time, the PV will eventually cross one of the outer alarm thresholds, named High-high alarm and Low-low alarm. Their threshold values are programmed using the loop table registers listed above. A High-high or Low-low alarm indicates a serious condition exists, and needs the immediate attention of the operator.

The PV Absolute Value Alarms are reported in the four bits in the PID Mode and Alarm Status word in the loop table, as shown to the right. We highly recommend using ladder logic to monitor these bits. The bit-of-word instructions make this easy to do. Additionally, you can monitor PID alarms using ***Direct***SOFT.

PID Mode and Alarm Status V+06





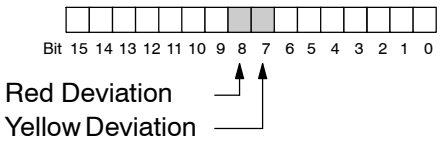
Loop Table

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 just outside the green zone, and the red zones are just beyond those.

The PV Deviation Alarms are reported in the two bits in the PID Mode and Alarm Status word in the loop table, as shown to the right. We highly recommend using ladder logic to monitor these bits. The bit-of-word instructions make this easy to do. Additionally, you can monitor PID alarms using ***Direct***SOFT.

PID Mode and Alarm Status V+06



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.

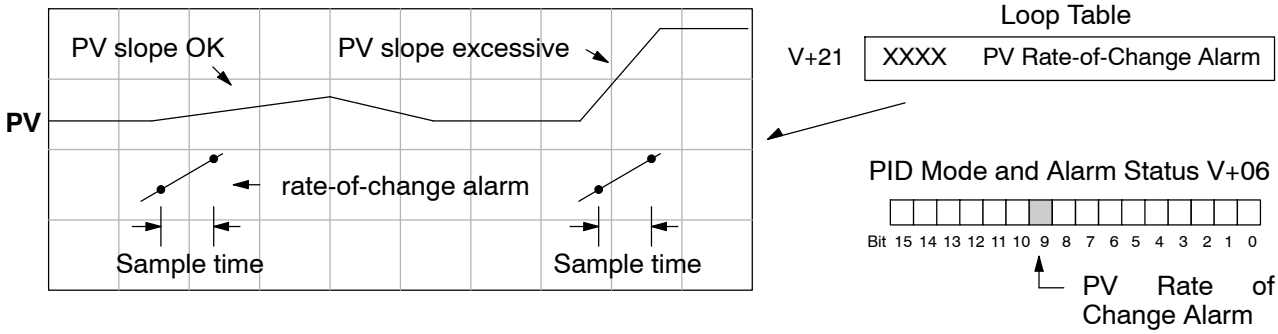
Remember, the alarm hysteresis feature works in conjunction with both the deviation and absolute value alarms, and is discussed at the end of this section.

See Errata Sheet at the beginning of this file.
Add the following note here:
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 DL450 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 temperature for our process, and we want an alarm when the temperature changes faster than 15 degrees/minute. We must know PV counts per degree and the loop sample rate. Then, suppose the PV value (in V+03 location) represents 10 counts per degree, and the loop sample rate is 2 seconds. We will use the formula below to convert our engineering units to counts/sample period:

$$\text{Alarm Rate-of-Change} = \frac{15 \text{ degrees}}{1 \text{ minute}} \times \frac{10 \text{ counts / degree}}{30 \text{ loop samples / min.}} = \frac{150}{30} = 5 \text{ counts / sample period}$$

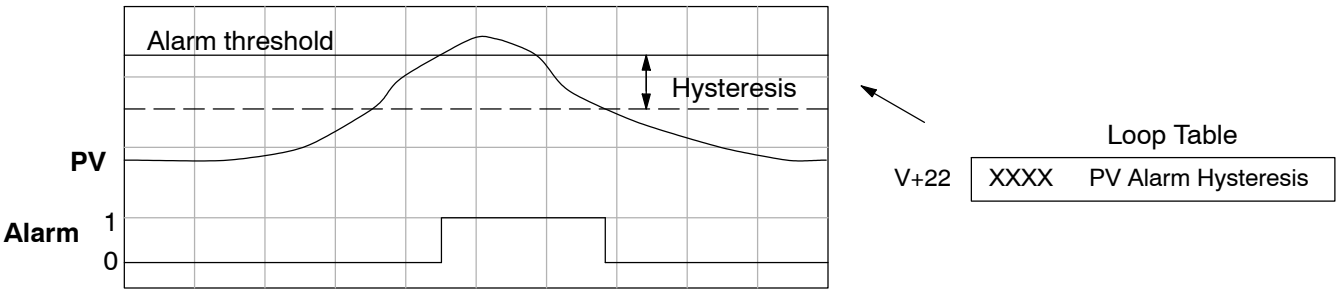
From the calculation result, we would program the value “5” in the loop table for the rate-of-change. The PV Rate-of-Change Alarm can be independently enabled and disabled from the other PV alarms, using bit 14 of the PID Mode 1 Setting V+00 word.

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.

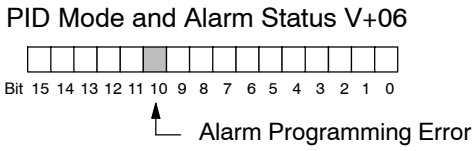
The PV Alarm Hysteresis amount is programmable from 1 to 200 (binary/decimal). When using the PV Deviation Alarm, the programmed hysteresis amount must be less than the programmed deviation amount. The figure below shows how the hysteresis is applied when the PV value goes past a threshold and descends 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

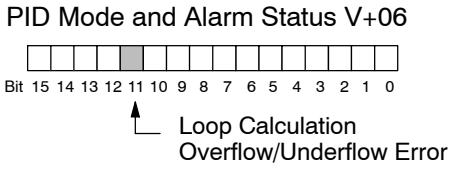
The PV Alarm threshold values must have certain mathematical relationships to be valid. The requirements are listed below. If not met, the Alarm Programming Error bit will be set, as indicated to the right.



- PV Absolute Alarm value requirements:
  Low-low < Low < High < High-high
- PV Deviation Alarm requirements:
  Yellow < Red

Loop Calculation Overflow/Underflow Error

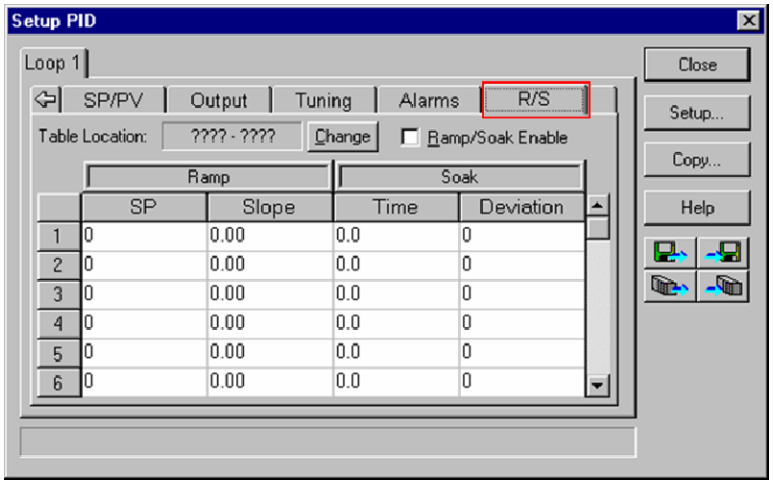
This error occurs whenever the output reaches it's upper or lower limit and the PV does not reach the setpoint. A typical example might be when a valve is stuck, the output is at it's limit, but the PV has not reached setpoint.



NOTE: Overflow/underflow can be alarmed in PID View. The optional C-more operator interface panel (see the automationdirect.com website) can also be setup to read these error bits using the PID Faceplate templates.

Ramp/Soak

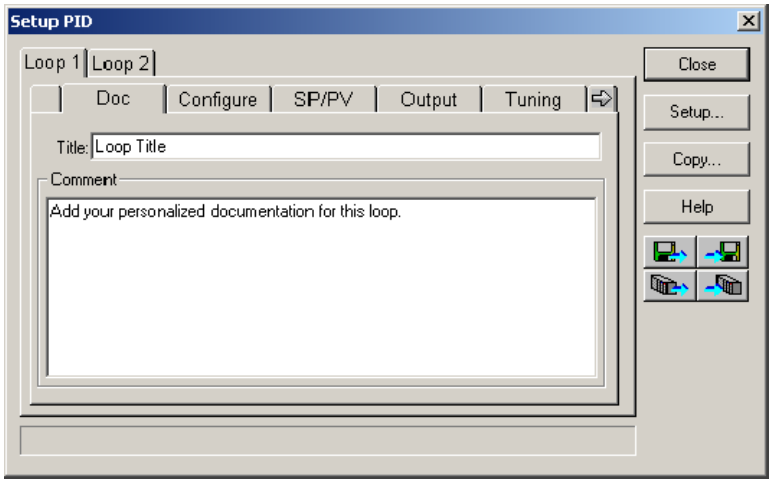
R/S (Ramp/Soak) is the last dialog available in the PID setup. The basic PID does not require any entries to be made in order to operate the PID loop. Ramp/Soak will be discussed in another section in this chapter.



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 are more than one PID loop in your application.



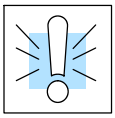
NOTE: It is good practice to save your project after setting up the PID loop by selecting **File** from the menu toolbar, then **Save project > to disk**. In addition to saving your entire project, all the PID parameters are also saved.

PID Loop Tuning

Once you have set up a PID loop, it must be tuned in order for it to work. The goal of loop tuning is to adjust the loop gains so the loop has optimal performance in dynamic conditions. The quality of a loop's performance may generally be judged by how well the PV follows the SP after 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 DL450 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 DL450 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 beginning on page 8-25). However, if you are going to use auto tuning, the auto tuning procedure will automatically calculate the sample rate in addition to the PID gains.

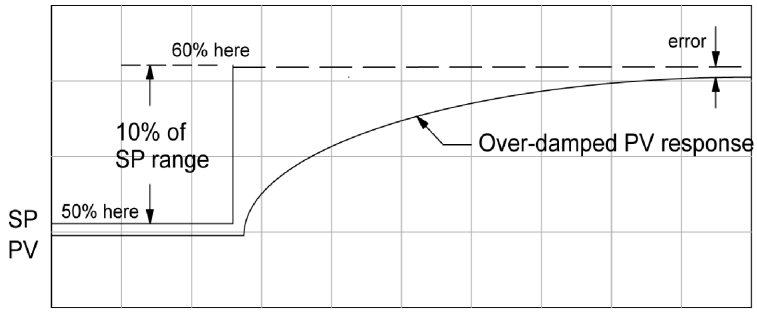
Manual Tuning Procedure

It is not necessary to try to obtain the best values for the P, I and D parameters in the PID loop by trial and error. Following is a typical procedure for tuning a temperature control loop which you may use to tune your loop.

Monitor the values of SP, PV and CV with a loop trending instrument or use the PID View feature in ***Direct***SOFT (see page 8-49).

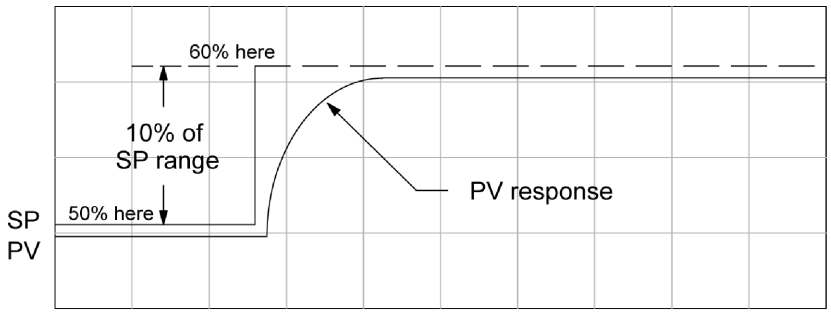
NOTE: We recommend using the PID View to select manual for the vertical scale feature, for both SP/PV area and Bias/Control Output areas. The auto scaling feature would otherwise change the vertical scale on the process parameters and add confusion to the loop tuning process.

- 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.

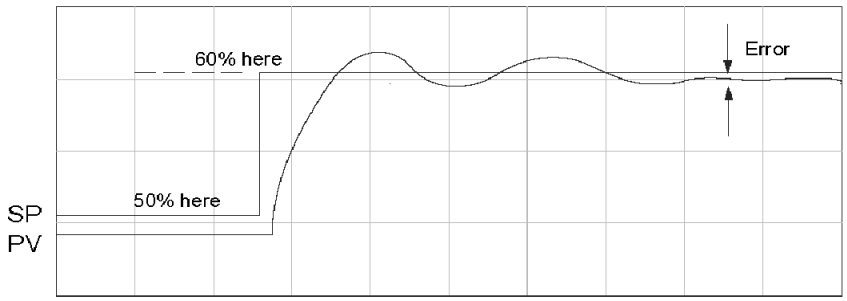


The response may take awhile, but you will see that there isn't any oscillation. This response is not desirable since it takes a long time to correct the error; also, there is a difference between the SP and the PV.

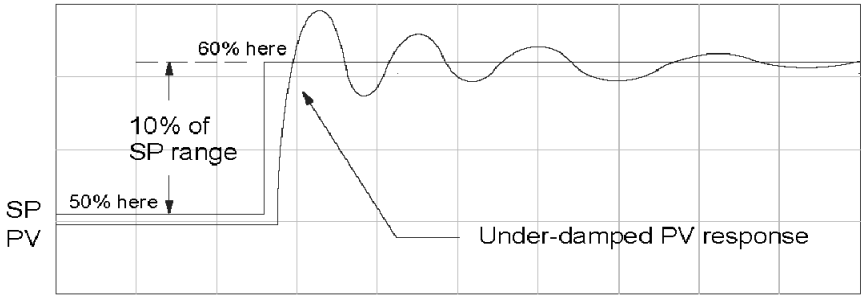
- Increase the Proportional gain, for example to 2.0. The control output will be greater and the response time will be quicker. The trend should resemble the figure below.



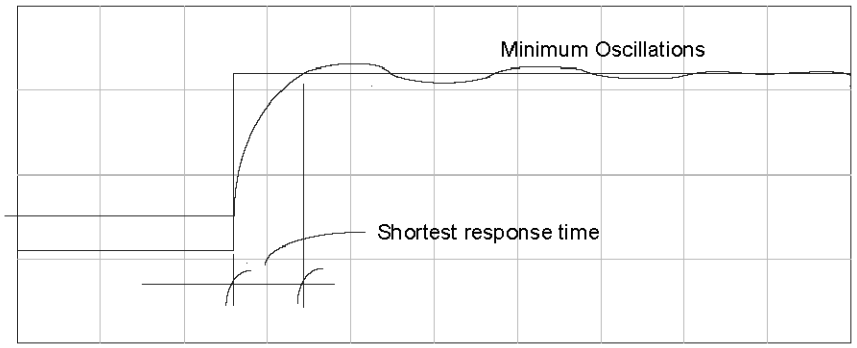
- Increase the Proportional gain in small increments, such as 4, 6, 7, etc. until the control output response begins to oscillate. This is the Proportional gain that should be recorded.



- 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.



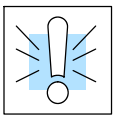
The foregone method is the most common method used to tune a PID loop. Derivative gain is almost never used in a temperature control loop. This method can also be used for other control loops, but other parameters may need to be added for a stable control output.

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.

Auto Tuning Procedure

The auto tuning feature for the DL450 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.

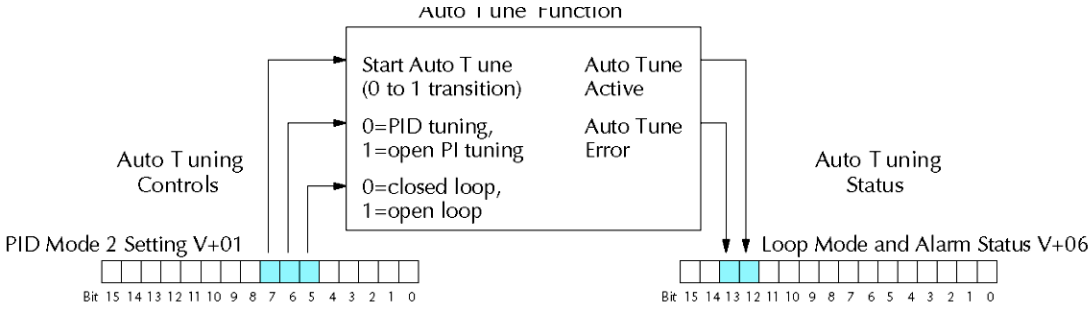


WARNING: Only authorized personnel fully familiar with all aspects of the process should make changes that affect the loop tuning constants. Using the loop auto tuning procedures will affect the process, including inducing large changes in the control output value. Make sure you thoroughly consider the impact of any changes to minimize the risk of injury to personnel or damage to equipment. The auto tune in the DL450 is not intended to be used as a replacement for your process knowledge.

Once the physical loop components are connected to the PLC, auto tuning can be initiated within ***Direct***SOFT (see the ***Direct***SOFT Programming Software Manual), and it can be used to establish initial PID parameter values. Auto tuning is the best "guess" the CPU can do after some trial tests.

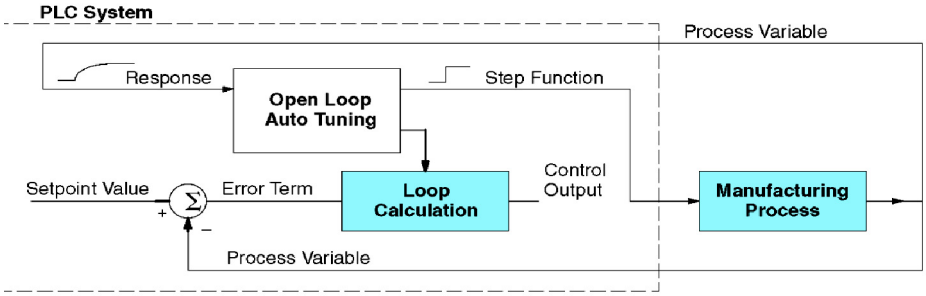
The loop controller offers both closed-loop and open-loop methods. The following sections describe how to use the auto tuning feature, and what occurs in open and closed-loop auto tuning.

The controls for the auto tuning function use three bits in the PID Mode 2 word V+01, as shown below. ***Direct***SOFT will manipulate these bits automatically when you use the auto tune feature within ***Direct***SOFT. Or, you may have your ladder logic access these bits directly for allowing control from another source such as a dedicated operator interface. The individual control bits allow you to start the auto tune procedure, select PID or PI tuning and select closed-loop or open-loop tuning. If you select PI tuning, the auto tune procedure leaves the derivative gain at 0. The Loop Mode and Alarm Status word V+06 reports the auto tune status as shown. Bit 12 will be on (1) during the auto tune cycle, automatically returning to off (0) when done.



Open-Loop Auto Tuning

During an open-loop auto tuning cycle, the loop controller operates as shown in the diagram below. Before starting this procedure, place the loop in Manual Mode and ensure the PV and control output values are in the middle of their ranges (away from the end points).

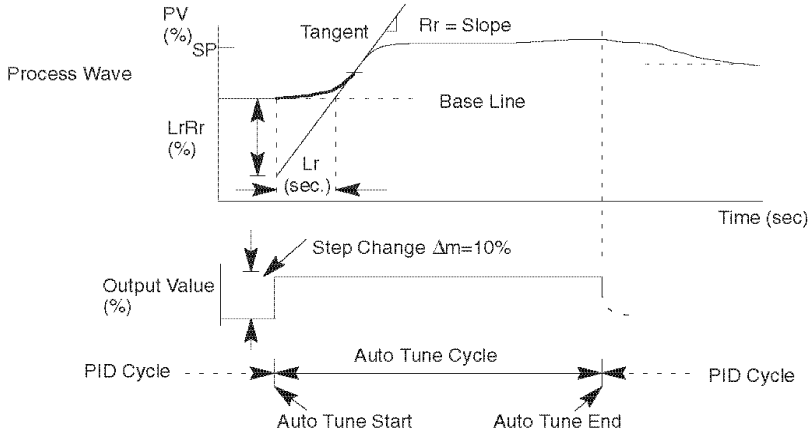


NOTE: In theory, the SP value does not matter in this case, because the loop is not closed. However, the requirement of the firmware is that the SP value must be more than 5% of the PV range from the actual PV before starting the auto tune cycle (for the DL450, 12 bit PV should be 205 counts or more 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.



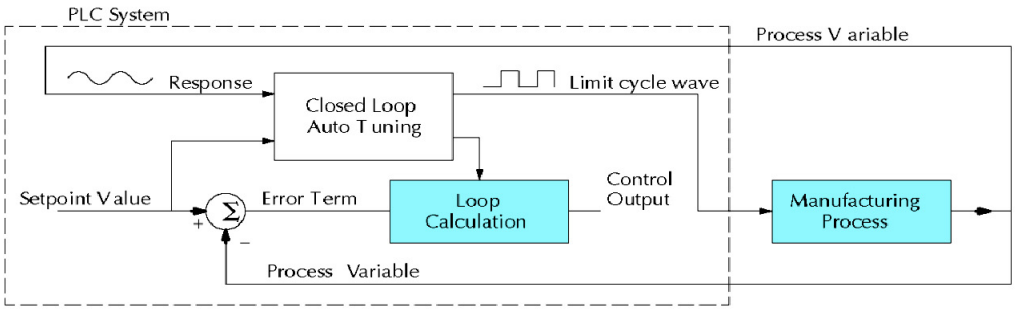
When the loop tuning observations are complete, the loop controller computes Rr (maximum slope in %/sec.) and Lr (dead time in sec). The auto tune function computes the gains according to the Zeigler-Nichols equations, shown below:

PID Tuning	SP Range
P = 1.2* ∆m/LrRr	P = 0.9* ∆m/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
∆m = Output step change (10% = 0.1, 20% = 0.2)	

We highly recommend using ***Direct***SOFT for the auto tuning interface. The duration of each auto tuning cycle will depend on the mass of the process. A slowly-changing PV will result in a longer auto tune cycle time. When the auto tuning is complete, the proportional, integral, and derivative gain values are automatically updated in loop table locations V+10, V+11, and V+12 respectively. The sample time in V+07 is also updated automatically. You can test the validity of the values the auto tuning procedure yields by measuring the closed-loop response of the PV to a step change in the output. The instructions on how to do this are in the section on the manual tuning procedure (located prior to this auto tuning section).

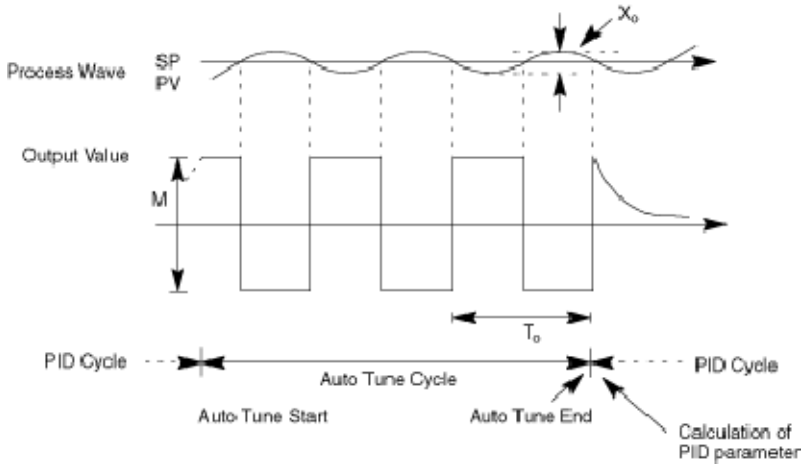
Closed-Loop Auto Tuning

During a closed-loop auto tuning cycle the loop controller operates as shown in the 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.



*Mmax = Output Value upper limit setting. Mmin = Output Value lower limit setting.

* This example is direct-acting.

When set to reverse-acting, the output will be inverted. When the loop tuning observations are complete, the loop controller computes T_o (bump period) and X_o (amplitude of the PV). Then it uses these values to compute Kpc (sensitive limit) and Tpc (period limit). From these values, the loop controller auto tune function computes the PID gains and the sample rate according to the Zeigler-Nichols equations shown below:

Kpc = 4M / (π * X0) Tpc = 0	
M = Amplitude of output	
PID Tuning	**PI Tuning**
P = 0.45 * Kpc	P = 0.30 * Kpc
I = 0.60 * Tpc	I = 1.00 * Tpc
D = 0.10 * Tpc	D = 0
Sample Rate = 0.014 * Tpc	Sample Rate = 0.03 * Tpc

Auto Tuning Error

In open-loop tuning, if the auto tune error bit (bit 13 of loop Mode/Alarm status word V+06) is on, please verify the PV and SP values are at least 5% of full scale difference, as required by the auto tune function.

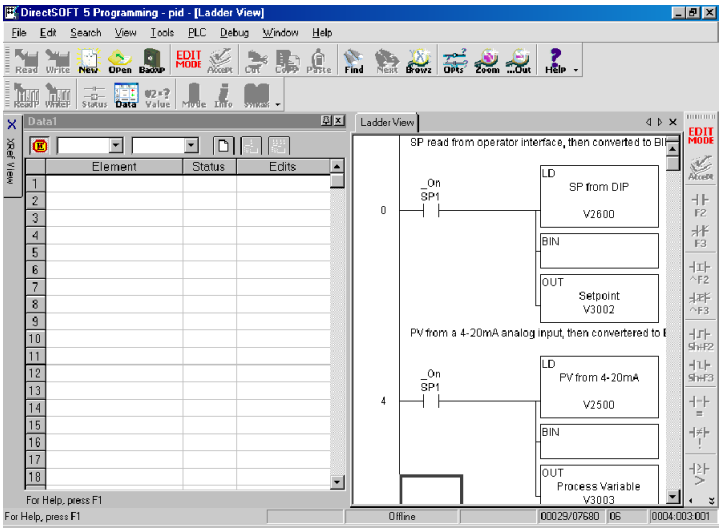
NOTE: If your PV fluctuates rapidly, you probably need to create a filter in ladder logic (see example on page 8-54).

Use *Direct*SOFT 5 Data View with PID View

The Data View window is a very useful tool which can be used to help tune your PID loop. You can compare the variables in the **PID View** with the actual values in the V-memory location with Data View.

Open a New Data View Window

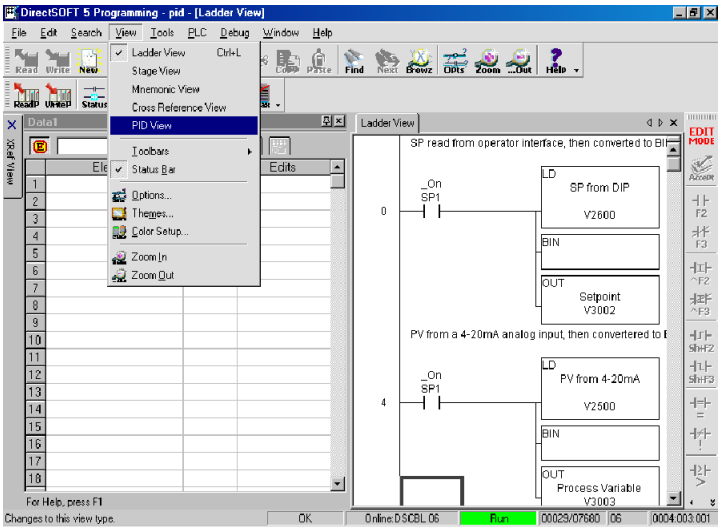
A new Data View window can be opened in any one of three ways; the menu bar **Debug > Data View > New**, the keyboard shortcut **Ctrl + Shift + F3** or the **Data** button on the Status toolbar. By default, the Data View window is assigned Data1 as the default name. This name can be changed for the current view using the Options dialog. The following diagram is an example of a newly opened Data View. The window will open next to the Ladder View by default.



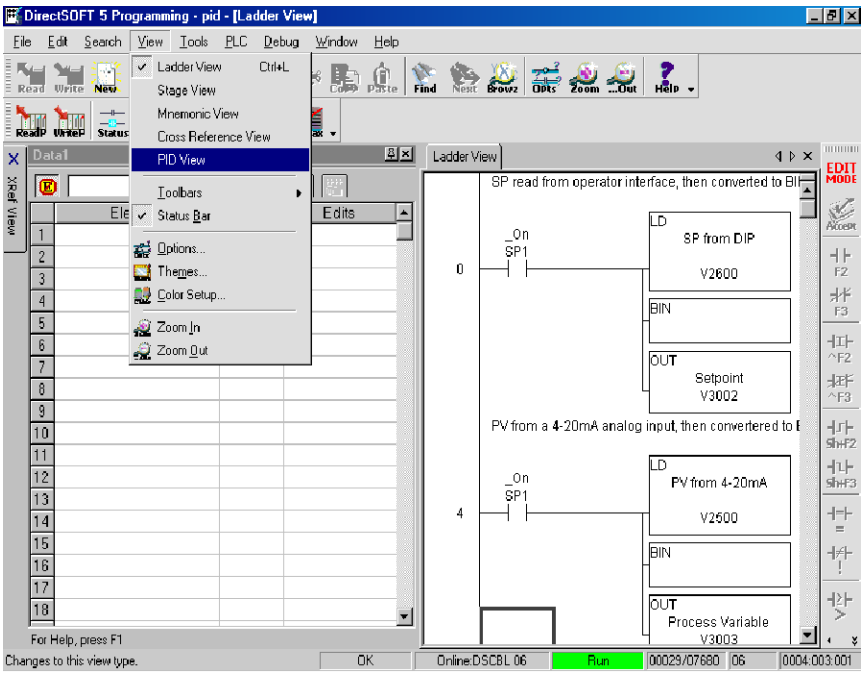
The Data View window can be used just as it is shown above for troubleshooting your PID logic, and it can be most useful when tuning the PID loop.

Open PID View

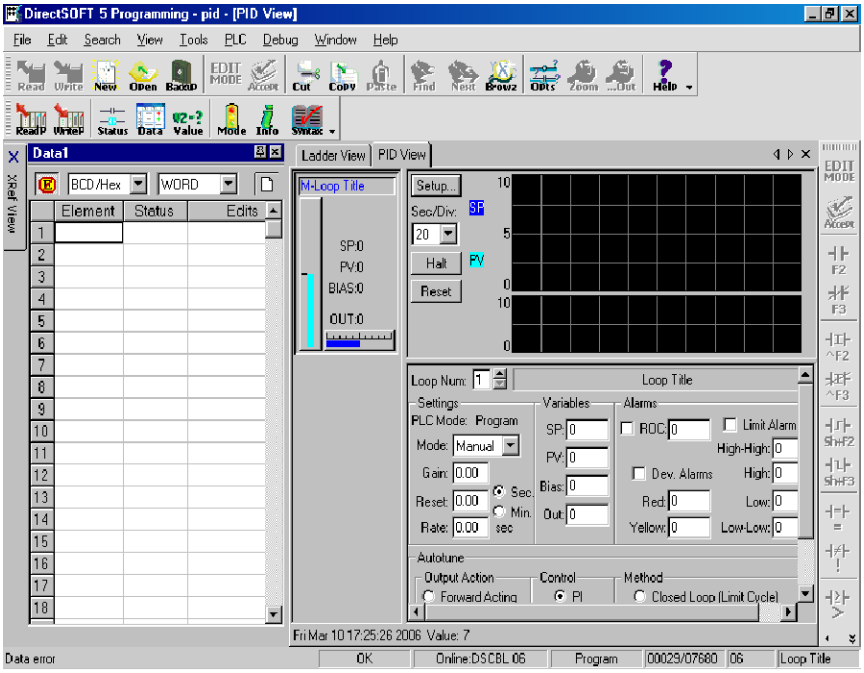
The PID View can only be opened after a loop has been setup in your ladder program and the programming computer is connected to the PLC (online). PID View is opened by selecting it from the View submenu on the Menu bar, **View > PID View**. The PID View can also be opened by clicking on the PID View button from the PLC Setup toolbar if it is in view.



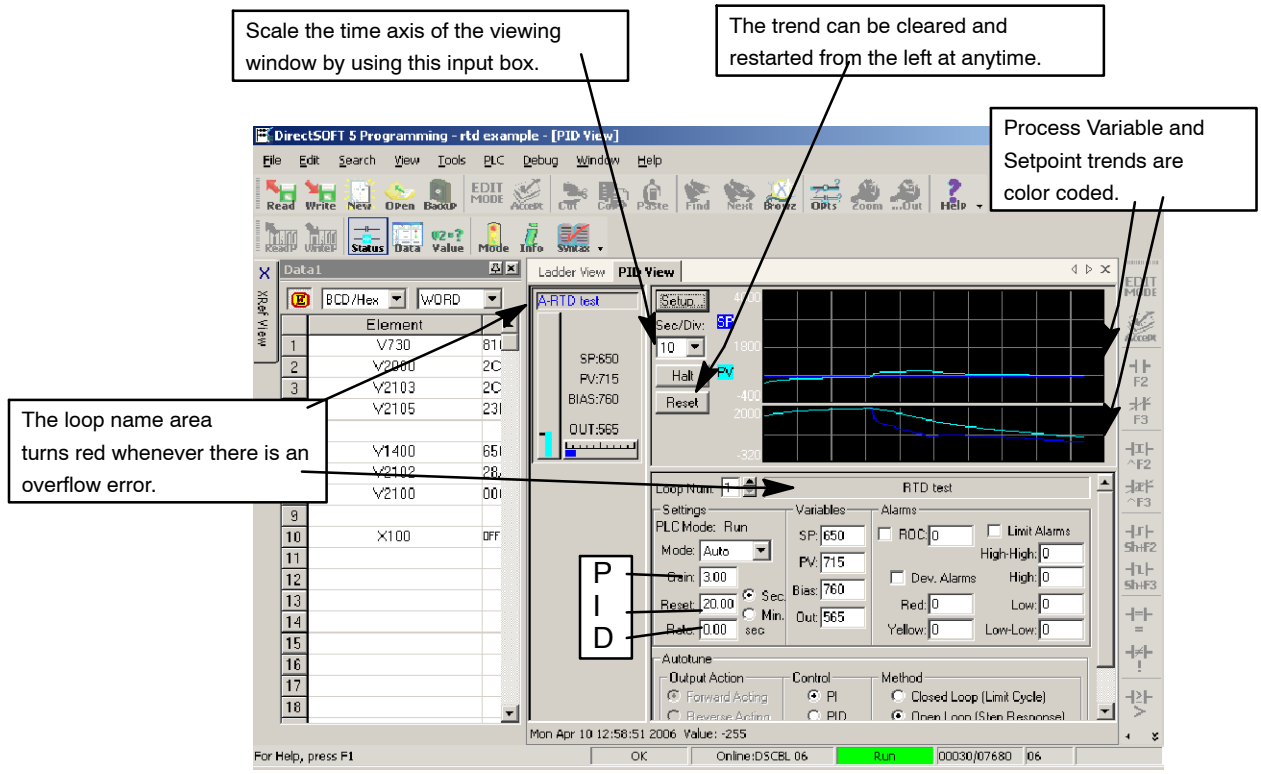
The PID View will open and appear over the Ladder View which can be brought into view by clicking on it's tab. When using the Data View and the PID View together, each view can be sized for better use as shown on the facing page.



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.



The diagram below illustrates how the 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.

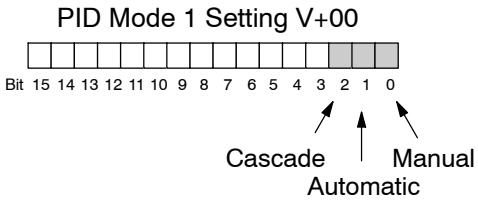


With both windows positioned in this manner, you are able to see where the PID values have been set and see the process that it is controlling.

Using Other PID Features

How to Change Loop Modes

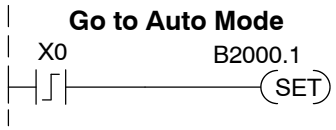
The first three bits of the PID Mode 1 word V+00 requests the operating mode of the corresponding loop. Note: These bits are mode change *requests*, not commands (certain conditions can prohibit a particular mode change – see next page).



The normal state of these mode request bits is "000". To request a mode change, you must SET the corresponding bit to a "1", for one scan. The PID loop controller automatically resets the bits back to "000" after it reads the mode change request. Methods of requesting mode changes are:

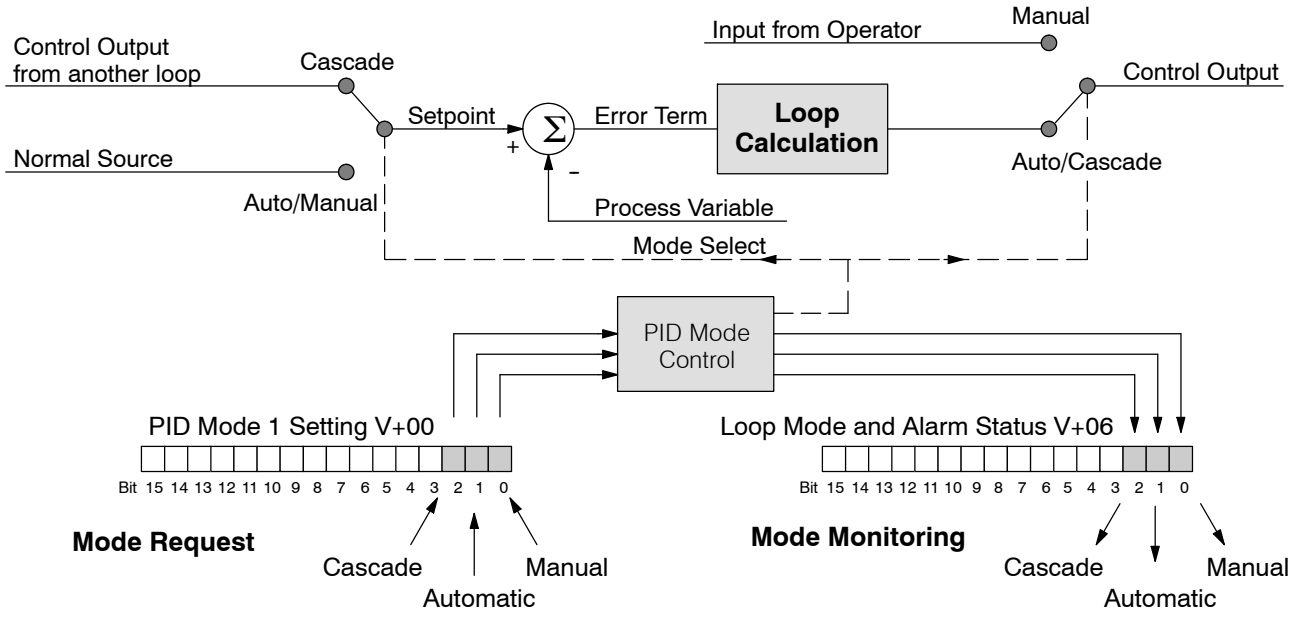
- ***Direct***SOFT's **PID View** – this is the easiest method. Use the drop–down menu, or click on one of the radio buttons if using older ***Direct***SOFT versions, and the appropriate bit will be set.
- **Ladder program**– ladder logic can request any loop mode when the PLC is in Run Mode. This will be necessary after application startup.

  Use the rung shown to the right to SET the mode bit on (do not use an out coil). On a 0–1 transition of X0, the rung sets the Auto bit = 1. The loop controller resets it.



- **Operator panel** – interface the operator's panel to ladder logic using standard methods, then use the technique above to set the mode bit.

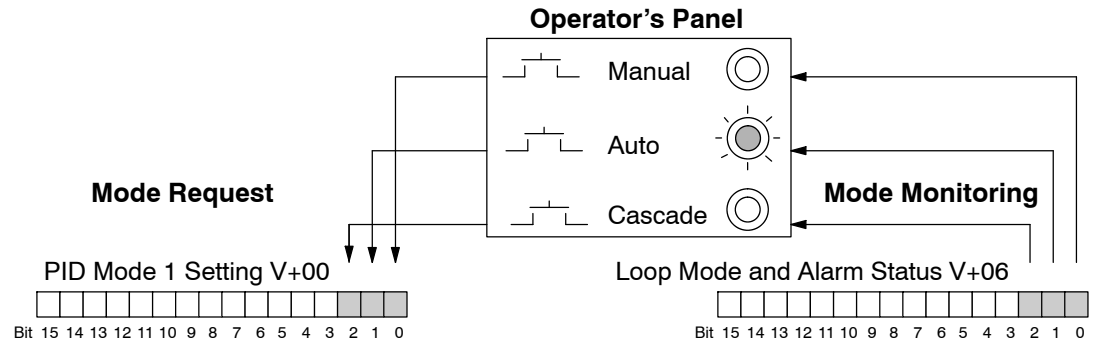
Since we can only *request* mode changes, the PID loop controller decides when to permit mode changes and provides the loop mode status. It reports the current mode on bits 0, 1, and 2 of the Loop Mode and Alarm Status word, location V+06 in the loop table. The parallel request / monitoring functions are shown in the figure below. The figure also shows the mode-dependent two possible 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 a few advanced applications). Remember that mode controls are really *mode request* bits, and the actual loop mode is indicated elsewhere.

The following figure shows an operator's panel using momentary push-buttons to request PID mode changes. The panel's mode indicators do not connect to the switches, but interface to the corresponding data locations.



PLC Modes' Effect on Loop Modes

The modes of the PLC (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 and during PLC Run Mode operation, the mode of a loop is determined by the request to V+00, bits 0, 1, and 2. However, a condition exists which will prevent a requested mode change from occurring:

- A major loop of a cascaded pair of loops cannot go from Manual to Auto until its minor loop is in Cascade mode.

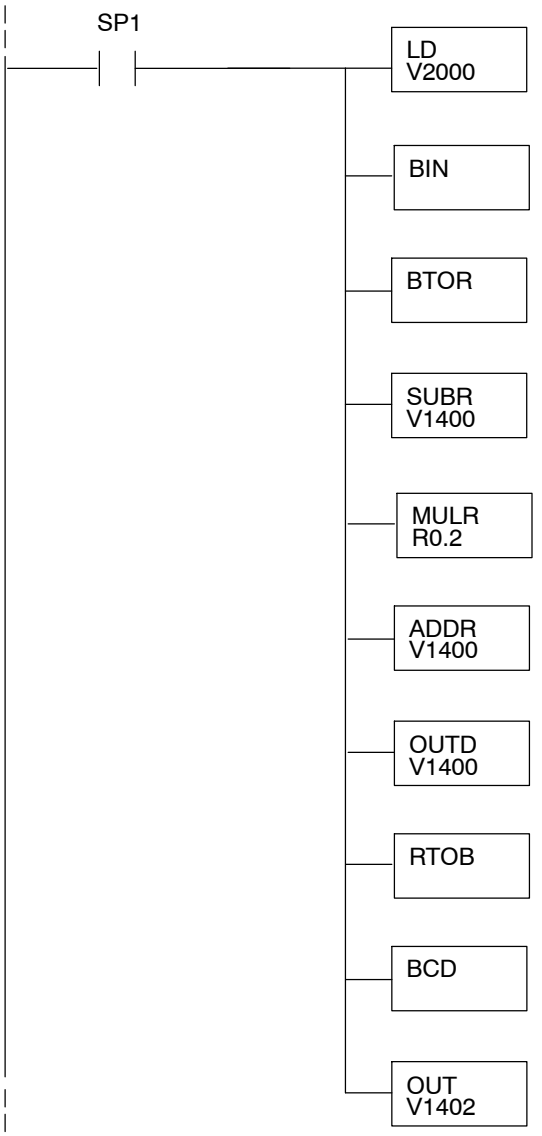
In other situations, the PID loop controller will automatically change the mode of the loop to ensure safe operation:

- 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.

Creating an Analog Filter in Ladder Logic

The DL450 does not support a built in filter, however, your analog inputs can be filtered effectively using the following RLL program example. 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”. 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 certain that 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.

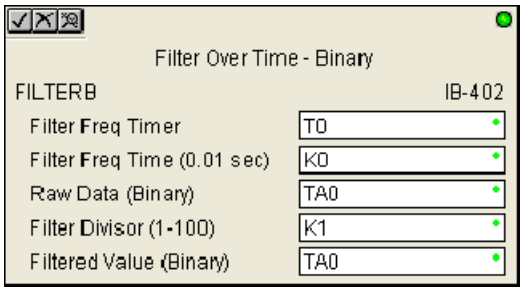
Use the *Direct*SOFT 5 Filter Intelligent Box Instruction

For those who are using ***Direct***SOFT 5, you have the opportunity to use the Analog Helper Intelligent Boxes (IBox) instructions. Following is one example which is available. IBox instruction IB-402, Filter Over Time in Binary (decimal) will perform a first-order filter on the Raw Data on a defined time interval. The equation is,

New = Old + [(Raw - Old)/FDC] where,

New = New Filtered Value
Old = Old Filtered Value
FDC = Filter Divisor Constant
Raw = Raw Data

The Filter Divisor Constant is an integer in the range K1 to K100, such that if it equaled K1, then no filtering is performed.



The rate at which the calculation is performed is specified by time in hundredths of a second (0.01 seconds) as the Filter Freq Time parameter. Note that this Timer instruction is embedded in the IBox and must NOT be used any other place in your program. Power flow controls whether the calculation is enabled. If it is disabled, the Filter Value is not updated. On the first scan from Program to Run mode, the Filter Value is initialized to 0 to give the calculation a consistent starting point.

Since the following binary filter example does not write directly to the PID PV location, the BCD filter could be used with BCD values and then converted to BIN.

FilterB Example

Following is an example of how the FilterB IBox is used in a ladder program. The instruction is used to filter a binary value that is in V2000. Timer (T1) is set to 0.5 seconds, the rate at which the filter calculation will be performed. The filter constant is set to 3.0. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering. The filtered value will be placed in V2100.

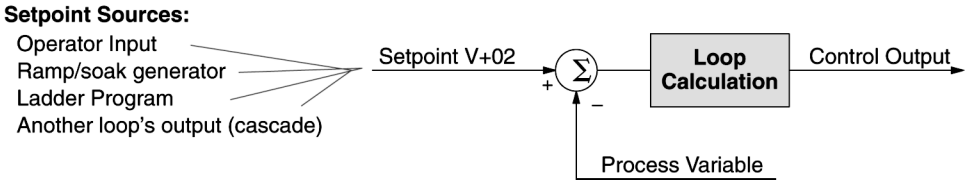
See DL405 IBox Instructions PLC User Manual Supplement for more detailed information.

NOTE: In order to use the IBox instructions found in ***Direct***SOFT 5, the D4-450 CPU must have firmware version 3.30 or higher installed.

Ramp/Soak Generator

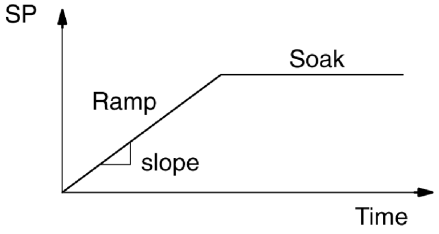
Introduction

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.



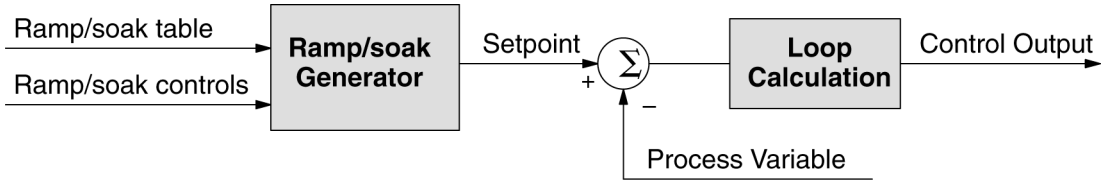
If the SP for your process rarely changes or can tolerate step changes, you probably will not need to use the ramp/soak generator. However, some processes require precisely-controlled SP value changes. The ramp/soak generator can greatly reduce the amount of programming required for these applications.

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. The soak time is also programmable in minutes.

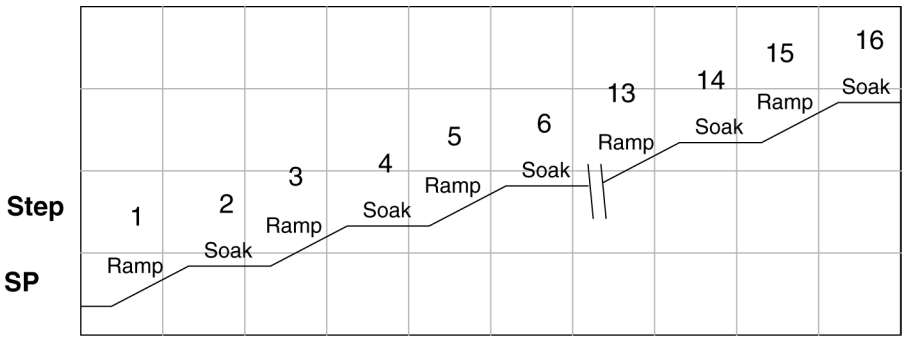
It is instructive to view the ramp/soak generator as a dedicated function to generate SP values, as shown below. It has two categories of inputs which determine the SP values generated. The ramp/soak table must be programmed in advance, containing the values that will define the ramp/soak profile. The 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 anytime 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.

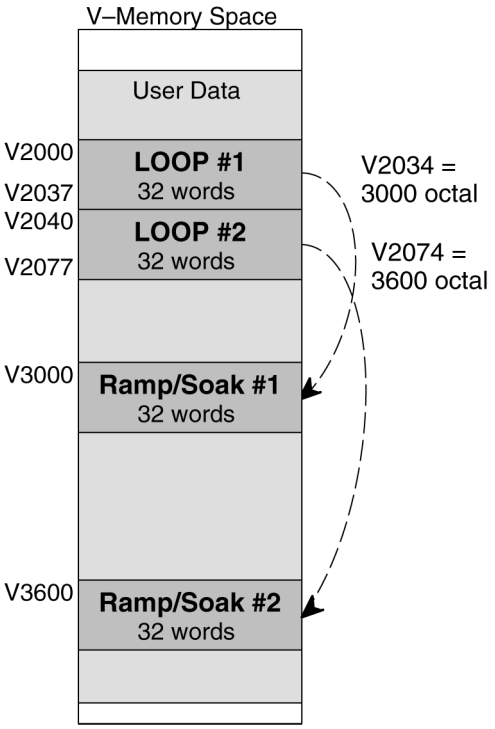
The following figure shows a SP profile consisting of ramp/soak segment pairs. The segments are individually numbered as steps from 1 to 16. The slope of each of the ramp may be either increasing or decreasing. The ramp/soak generator automatically knows whether to increase or decrease the SP based on the relative values of a ramp's end points. These values come from the ramp/soak table.



Ramp/Soak Table

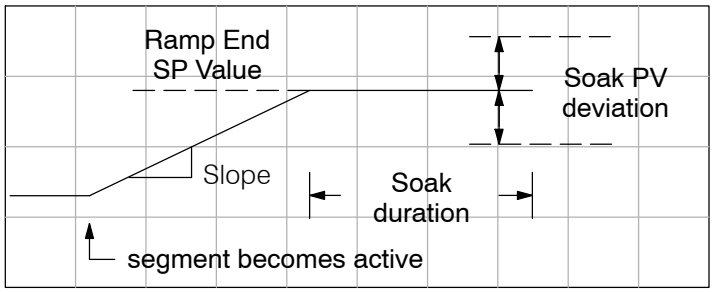
The parameters which define the ramp/soak profile for a loop are in a ramp/soak table. Each loop may have its own ramp/soak table, but it is optional. Recall the Loop Parameter table consists a 32-word block of memory for each loop, and together they occupy one contiguous memory area. However, the ramp/soak table for a loop is individually located, because it is optional for each loop. An address pointer in location V+34 in loop table specifies the starting location of the ramp/soak table.

In the example to the right, the loop parameter tables for Loop #1 and #2 occupy contiguous 32-word blocks as shown. Each has a pointer to its ramp/soak table, independently located elsewhere in user V-memory. Of course, you may locate all the tables in one group, as long as they do not overlap.



The parameters in the ramp/soak table must be user-defined. the most convenient way is to use ***Direct***SOFT, which features a special editor for this table. Four parameters are required to define a ramp and soak segment pair, as pictured below.

- **Ramp End Value** — specifies the destination SP value for the end of the ramp. Use the same data format for this number as you use for the SP. It may be above or below the beginning SP value, so the slope could be up or down (we don't have to know the starting SP value for ramp #1).
- **Ramp Slope** — specifies the SP increase in counts (units) per second. It is a BCD number from 00.00 to 99.99 (uses implied decimal point).
- **Soak Duration** — specifies the time for the soak segment in minutes, ranging from 000.1 to 999.9 minutes in BCD (implied decimal point).
- **Soak PV Deviation** — (optional) specifies an allowable PV deviation above and below the SP value during the soak period. A PV deviation alarm status bit is generated by the ramp/soak generator.



Ramp/Soak Table

V+00	XXXX	Ramp End SP Value
V+01	XXXX	Ramp Slope
V+02	XXXX	Soak Duration
V+03	XXXX	Soak PV Deviation

The ramp segment becomes active when the previous soak segment ends. If the ramp is the first segment, it becomes active when the ramp/soak generator is started, and automatically assumes the present SP as the starting SP.

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

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.

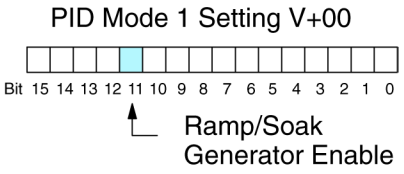
Ramp/Soak Table Flags

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)	

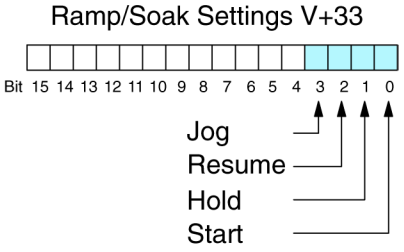
Ramp/Soak Generator Enable

The main enable control to permit ramp/soak generation of the SP value is accomplished with bits 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

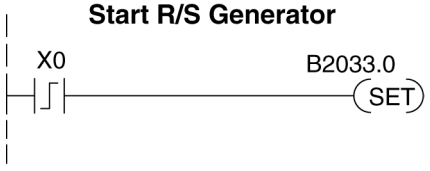
The four main controls for the ramp/soak generator are in bits 0 to 3 of the ramp/soak settings word in the loop parameter table. ***Direct***SOFT controls these bits directly from the ramp/soak settings dialog. However, you must use ladder logic to control these bits during program execution. We recommend using the bit-of-word instructions.



Ladder logic must set a control bit to a "1" to command the corresponding function. When the loop controller reads the ramp/soak value, it automatically turns off the bit for you. Therefore, a reset of the bit is not required when the CPU is in Run Mode.

The example program rung to the right shows how an external switch X0 can turn on and the PD contact uses the leading edge to set the proper control bit to start the ramp soak profile. This uses the Set Bit-of-Word instruction.

Start R/S Generator



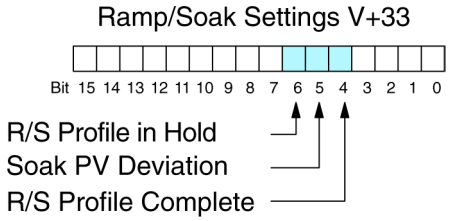
The normal state for the ramp/soak control bits is all zeros. Ladder logic must set only one control bit at a time.

- **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 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.

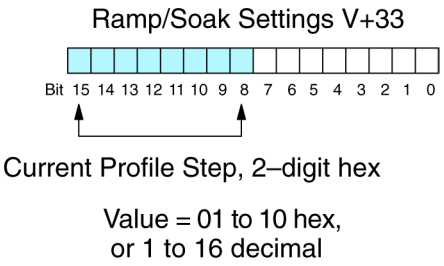
Ramp/Soak Profile Monitoring

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 pro file was active but is now in hold. Ramp/ Soak Settings V+33.

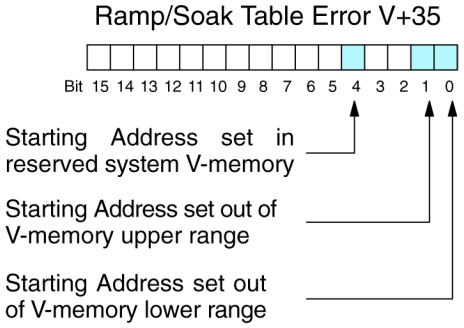


The number of the current step is available in the upper 8 bits of the Ramp/Soak Settings V+33 word. The bits represent a 2-digit hex number, ranging from 1 to 10. Ladder logic can monitor these to synchronize other parts of the program with the ramp/soak profile. Load this word to the accumulator and shift right 8 bits, and you have the step number.



Ramp/Soak Programming Errors

The starting address for the ramp/soak table must be a valid location. If the address points outside the range of user V-memory, one of the bits to the right will turn on when the ramp/soak generator is started. We recommend using ***Direct***SOFT to configure the ramp/soak table. It automatically range checks the addresses for you.



Testing Your Ramp/Soak Profile

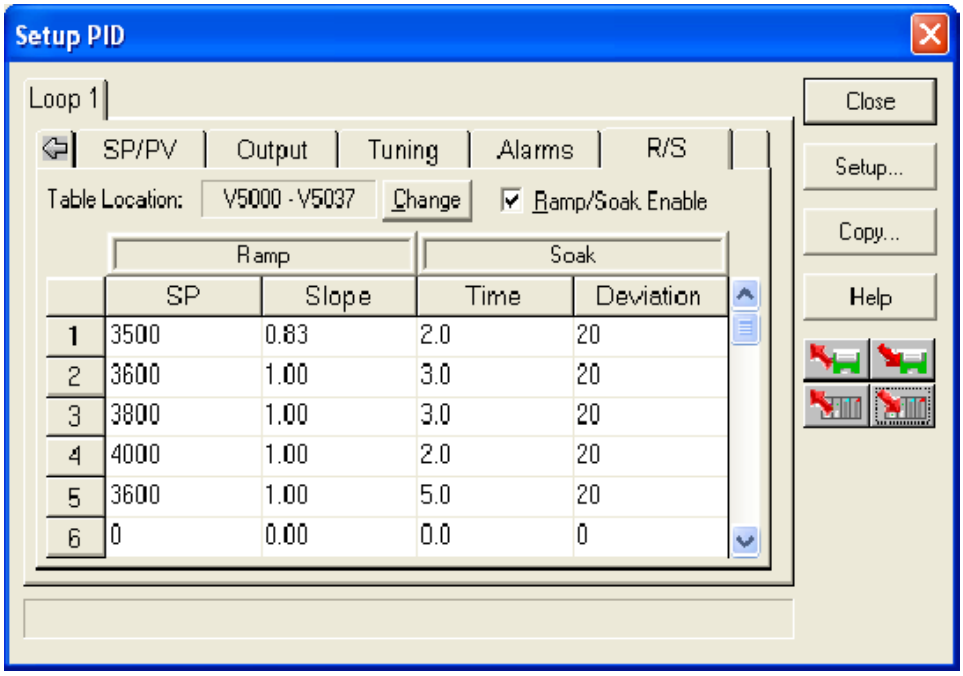
It's a good idea to test your ramp/soak profile before using it to control the process. This is easy to do, because the ramp/soak generator will run even when the loop is in Manual Mode. Using ***Direct***SOFT's PID View will be a real time-saver, because it will draw the profile on-screen for you. Be sure to set the trending timebase slow enough to display completed ramp-soak segment pairs in the waveform window.

*Direct*SOFT Ramp/Soak Example

The following example will step you through the Ramp/Soak setup.

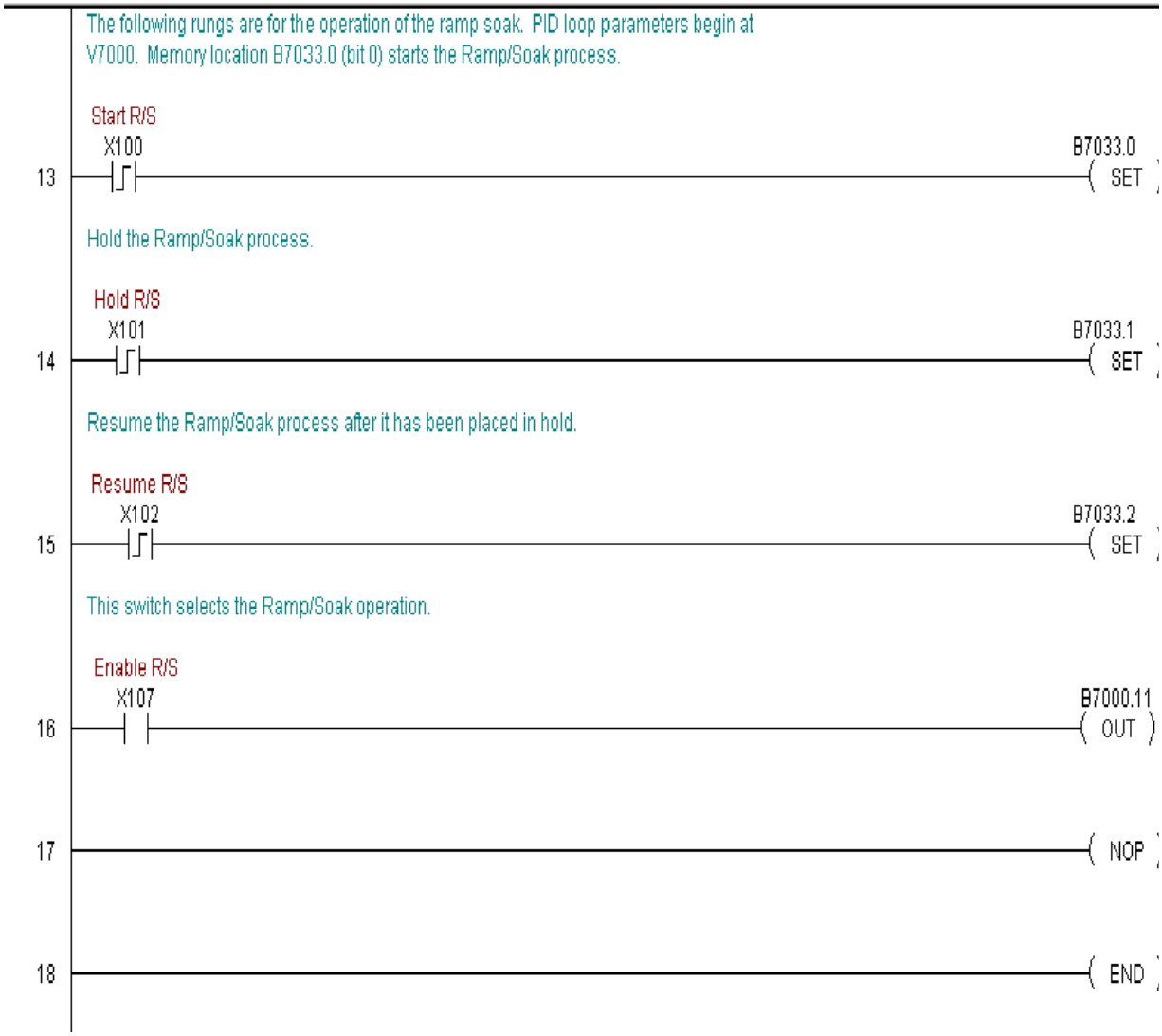
Setup the Profile in PID Setup

The first step is to use Setup in ***Direct***SOFT PID to set the profile of your process. Open the Setup PID window and select the R/S tab, and then enter the Ramp/Soak data. Note the V-memory location for the beginning of this profile is V5000, and V5037 is the end of the range of the profile.



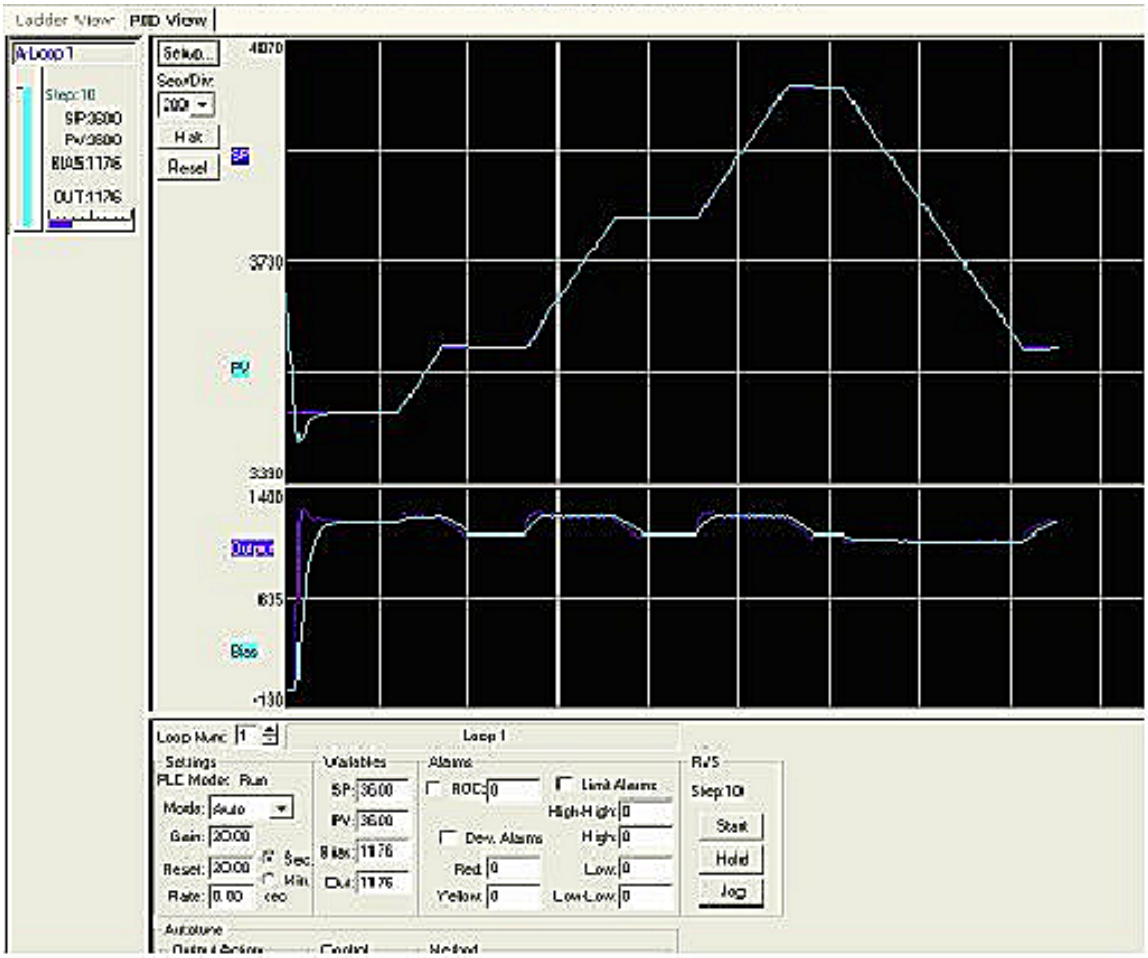
Program the Ramp/Soak Control in Relay Ladder

Refer to the Ramp/Soak Flag Bit Description table on page 8-59 when adding the control rungs to your program similar to the ladder rungs below. For the example below, the PID parameters begin at V7000. The Ramp/Soak bit flags are located at V7033.



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.



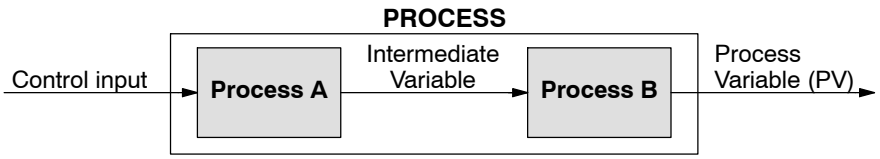
Cascade Control

Introduction

Cascaded loops are an advanced control technique that is 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 DL450 also provides Cascaded Mode.

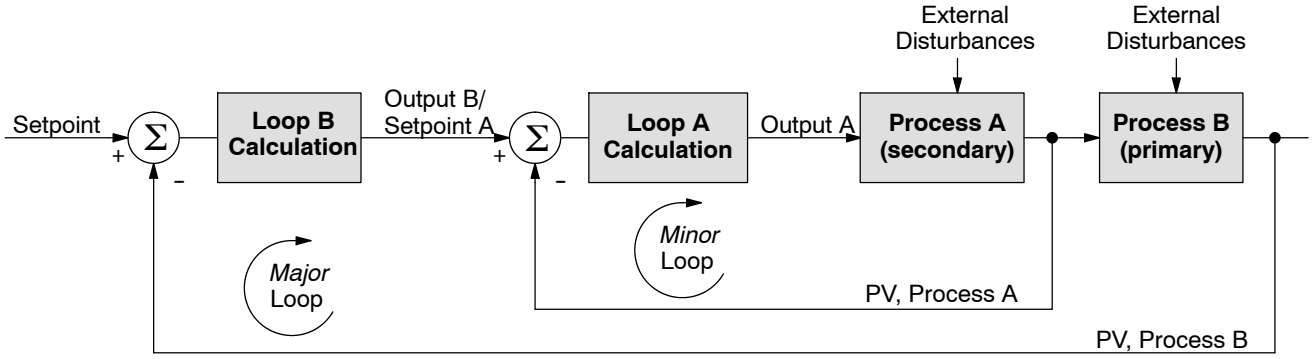
NOTE: Cascaded loops are an advanced process control technique. Therefore, we recommend their use only for experienced process control engineers.

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 that 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. We do have to add the additional sensor to measure the intermediate variable (PV for process A). Notice that 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 that 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.

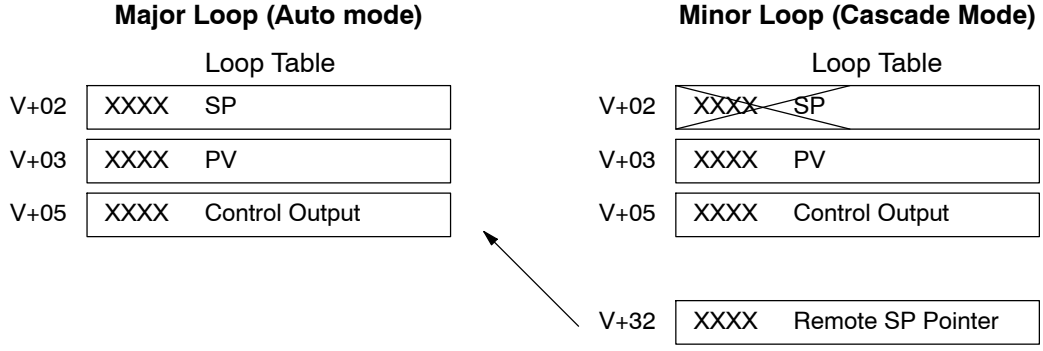
Cascaded Loops in the DL450 CPU

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. Just remember that all minor loops will be in Cascade Mode, and only the outer-most (major) loop will be in Auto Mode.

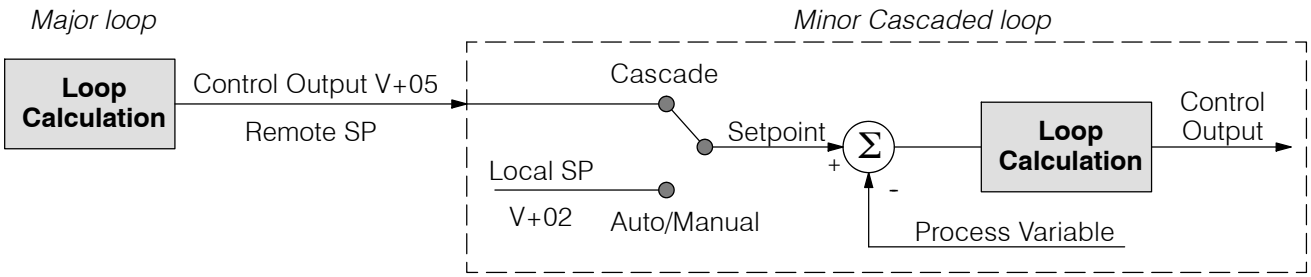
You can cascade together as many loops as necessary on the DL450, and you may have multiple groups of cascaded loops. For proper operation on cascaded loops you must use the same data range (12/15 bit) and polar/bipolar settings on the major and minor loop.

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 the its local SP register (V+02), and read the major loop's control output as its SP instead.



When using ***Direct***SOFT's PID View to watch the SP value of the minor loop, ***Direct***SOFT automatically reads the major loop's control output and displays it for the minor loop's SP. The minor loop's normal SP location, V+02, remains unchanged.

Now, we use the loop parameter arrangement above and draw its equivalent loop schematic, 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

When tuning cascaded loops, you will need to de-couple the cascade relationship and tune the minor loop, using one of the loop tuning procedures previously covered. Once this has been done, have the minor loop in cascade mode and auto tune the major loop (see Step 4).

1. If you are not using auto tuning, then find the loop sample rate for the minor loop, using the method discussed earlier in this chapter. Then set the sample rate of the major loop slower than the minor loop by a factor of 10. Use this as a starting point.

2. Tune the minor loop first. Leave the major loop in Manual Mode, and you will need to generate SP changes for the minor loop manually as described in the loop tuning procedure.

3. Verify the minor loop gives a critically-damped response to a 10% SP change while in Auto Mode. Then we are finished tuning the minor loop.

4. In this step, you will need to get the minor loop in Cascade Mode, and then the Major loop in Auto Mode. We will be tuning the major loop with the minor loop treated as a series component its overall process. Therefore, do not go back and tune the minor loop again while tuning the major loop.

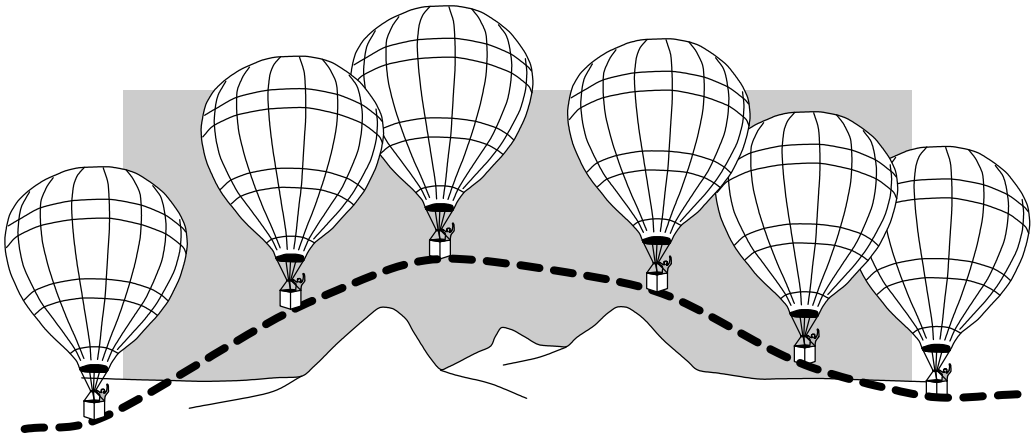
5. Tune the major loop, following the standard loop tuning procedure in this section. The response of the major loop PV is actually the overall response of the cascaded loops together.

Time-Proportioning Control

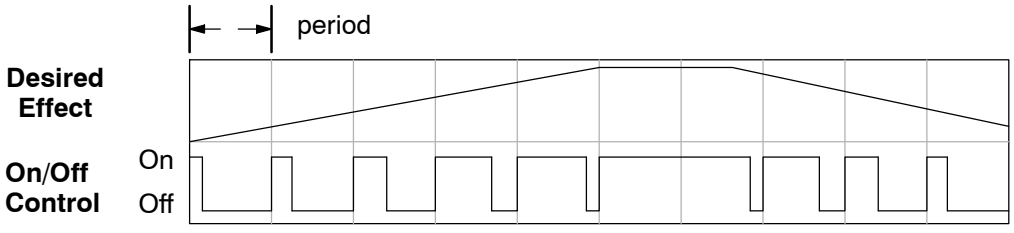
The PID loop controller in the DL450 CPU generates a smooth control output signal across a numerical range. The control output value is suitable to drive an analog output module, which connects to the process. In the process control field, this is called *continuous control*, because the output is on (at some level) continuously.

While continuous control can be smooth and robust, the cost of the loop components (such as actuators, 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*.



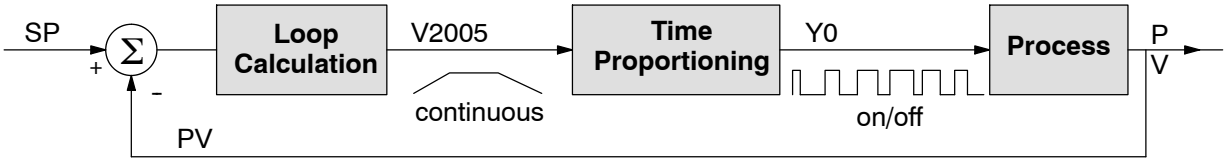
Time-proportioning control approximates continuous control by virtue of its duty-cycle – the ratio of ON time to OFF time. The following figure shows an example of how duty cycle approximates a continuous level when it is averaged by a large process mass.



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 ouptut coil, Y0.

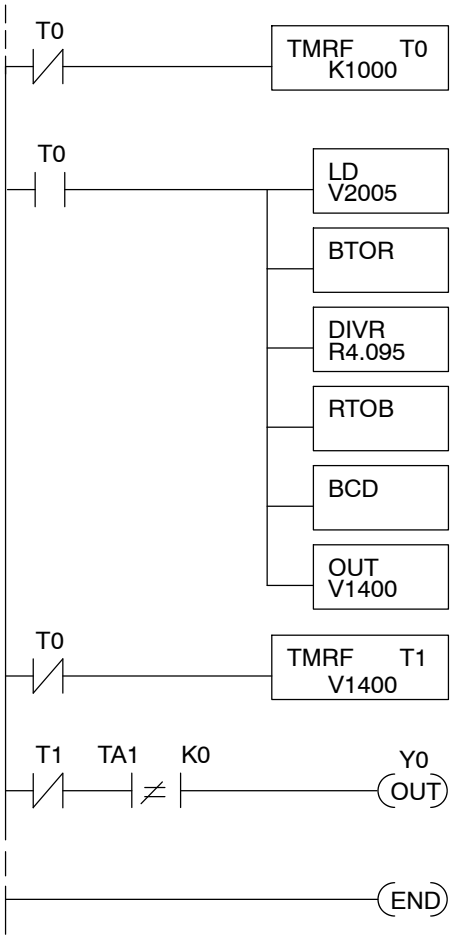


The example program uses two timers to generate on/off control. It makes the following **assumptions**, which you can alter to fit your application:

- 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 on/off control output is Y0.

The control program must "match" the resolution of the PID output to the resolution of the time interval. The time interval for one full cycle of the on/off waveform is 10 seconds.

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.



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 preset time for TMRF T0.

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 TMRF T1.

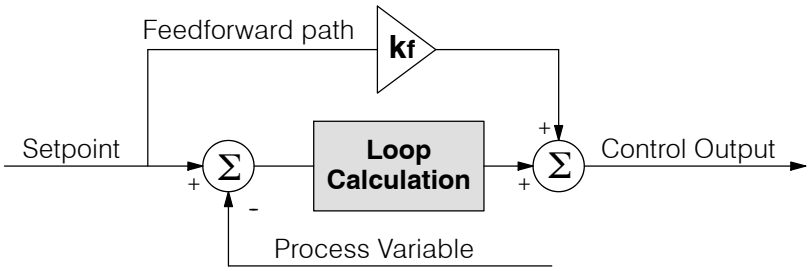
The second fast timer also counts in increments of .01 seconds, so its range is variable from 0 to a maximum of 1000 ticks, or 10 seconds. This timer's output, T1, turns off the output coil, Y0, when the preset is reached.

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 DL450. 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 "feed-forward" 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.

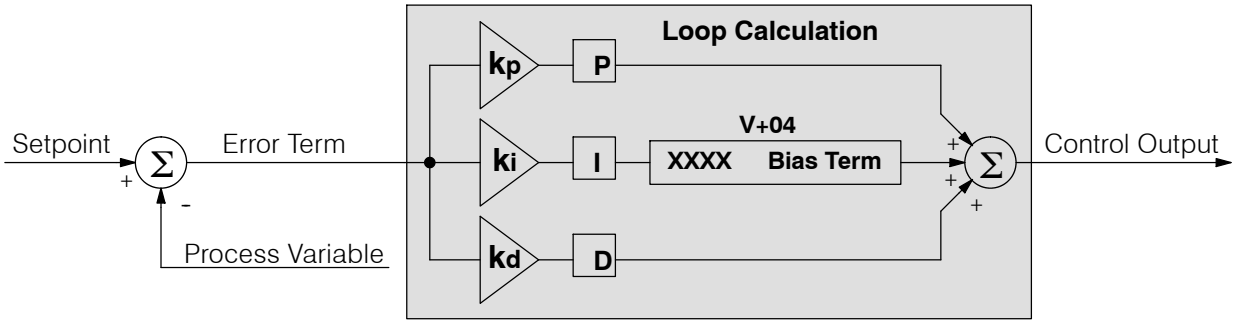


In the previous section on the bias term, we said that "the bias term value establishes a "working region" or operating point for the control output. *When the error fluctuates around its zero point, the output fluctuates around the bias value.*" Now, when there is a change in setpoint, an error is generated 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 from 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 DL450 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.

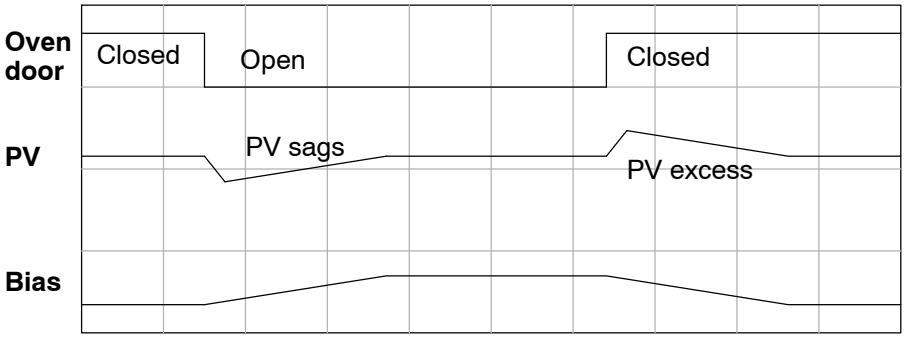


To change the bias (operating point), ladder logic only has to write the desired value to V+04. The PID loop calculation first reads the bias value from V+04 and modifies the value based on the current integrator calculation. Then it writes the result back to location V+04. This arrangement creates a 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 (the example below shows you how).

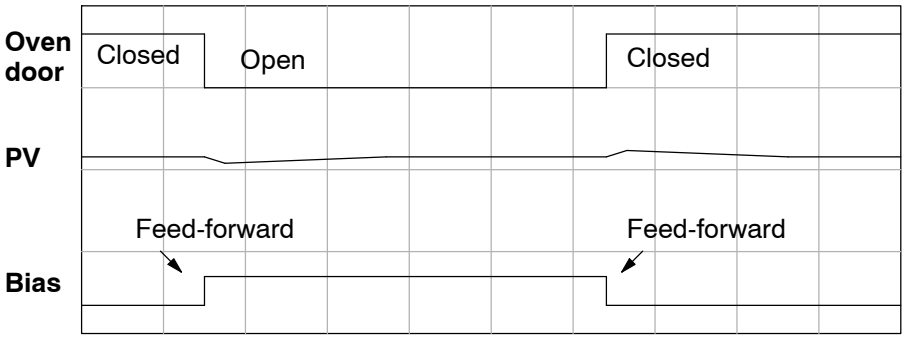
NOTE: When writing the bias term, one must be careful to design ladder logic to write the value just once, at the moment when the new bias operating point is to occur. If ladder logic writes the bias value on every scan, the loop’s integrator is effectively disabled.

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.

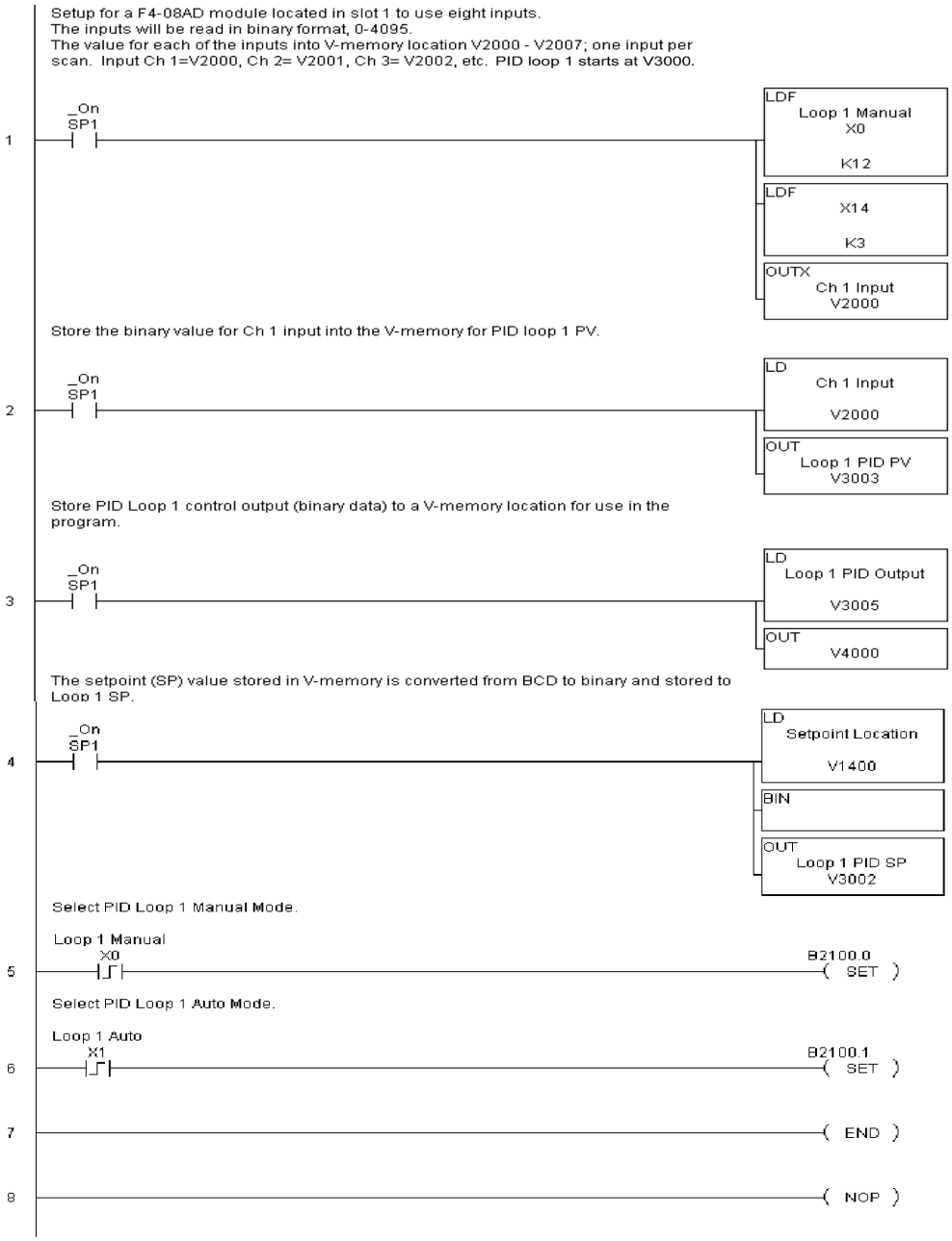


The step changes in the bias are the result of our two feed-forward writes to the bias term. We can see that the PV variations are greatly reduced. The same technique may be applied for changes in setpoint.

PID Example Program

Program Setup for the PID Loop

After the PID loop(s) has been setup with ***Direct***SOFT, 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 of the analog I/O modules used with the DL450 is setup in a similar manner. Refer to the DL405 Analog I/O Manual for the setup information for the particular module that you will be using.

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 ***Direct***SOFT Programming Software Manual provides more information for the use of PID View.

Troubleshooting Tips

Q. The loop will not go into Automatic Mode.

A. Check the following for possible causes:

- The PLC is in Program Mode. It must be in Run Mode for loops to run.
- 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 just 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 cause:

- The gain values are entered improperly. Remember, gains are entered in the loop table in BCD, while the SP and PV are in binary. If you are using ***Direct***SOFT 5, PID View displays the SP, PV, Bias and Control output in decimal, 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 that 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 that the analog module is generating the value, and that 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 Mode 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 *Direct*SOFT work okay, but these values do not work properly when the ladder program writes the data.

A. The PID View in ***Direct***SOFT lets you enter SP, PV, and Bias values in decimal, and displays them in decimal for your convenience. For example, when the data format is 12 bit unipolar, the values range from 0 to 4095. However, the loop table actually requires these in hex, so ***Direct***SOFT converts them for you. The values in the table range from 0 to FFF, for 12-bit unipolar format. 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 by increasing the integral time value, and the proportional gain if necessary.
- There is too much transfer lag in your process. This means the PV reacts sluggishly to control output changes. There may be too much “distance” between actuator and PV sensor, or the actuator may be weak in its ability to transfer energy into the process.
- There may be a process disturbance that is over-powering the loop. Make sure the PV is relatively steady when the SP is not changing.

Glossary of PID Loop Terminology

Automatic Mode	An operational mode of a loop, in which it makes PID calculations and updates the loop's control output.
Bias Freeze	A method of preserving the bias value (operating point) for a control output, by inhibiting the integrator when the output goes out-of-range. The benefit is a faster loop recovery.
Bias Term	In the position form of the PID equation, it is the sum of the integrator and the initial control output value.
Bumpless Transfer	A method of changing the operation mode of a loop while avoiding the usual sudden change in control output level. This consequence is avoided by artificially making the SP and PV equal, or the bias term and control output equal at the moment of mode change.
Cascaded Loops	A cascaded loop receives its setpoint from the output of another loop. Cascaded loops have a major/minor relationship, and work together to ultimately control one PV.
Cascade Mode	An operational mode of a loop, in which it receives its SP from another loop's output.
Continuous Control	Control of a process done by delivering a smooth (analog) signal as the control output.
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.
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.
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 DL450'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. Bryant 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

9

Maintenance and Troubleshooting

In This Chapter. . . .

- Hardware Maintenance
- Diagnostics
- CPU Indicators
- I/O Module Troubleshooting
- Noise Troubleshooting
- Machine Startup and Program Troubleshooting

Hardware Maintenance

Standard Maintenance

A routine maintenance check (about every one or two months) or your PLC and control system is good practice, an should include the following items:

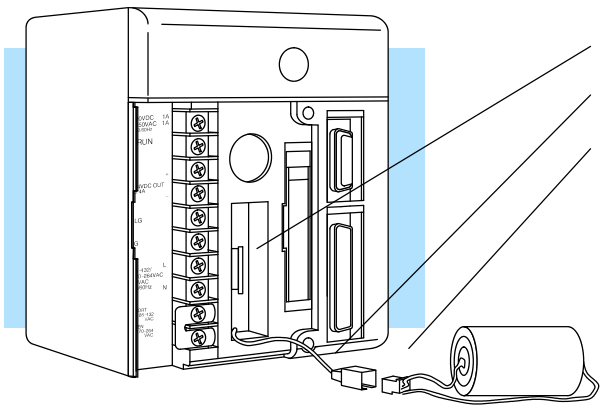
- **Air Temperature** – Check the ambient air temperature in the control cabinet, so the operating temperature range of any component is not exceeded.
- **Air Filter** – If the control cabinet has an air filter, clean or replace it periodically as required.
- **Memory Backup Battery** – Make sure the CPU memory backup battery does not become completely discharged. The CPU uses the same battery indicator to signal a low CPU battery and if applicable a low memory cartridge battery. Make sure you have correctly diagnosed which battery is requiring replacement.
- **Fuses or breakers** – verify that all fuses and breakers are intact.
- **DL405 Module Air Vents** – verify that all air vents of the CPU and all I/O modules are clear, allowing air to circulate for cooling.

CPU Battery Replacement

The CPU battery is used to retain program V-memory and the system parameters. The life expectancy of this battery is five years.

NOTE: Before replacing your CPU battery you should back-up your V-memory and system parameters. This can be done by saving the V-memory and system parameters to either a Memory Cartridge or to cassette tape or by using ***Direct***SOFT to save the program to hard/floppy disk on a personal computer.

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 is done to ensure the capacitor used to maintain the proper voltage levels necessary to retain memory is fully charged.



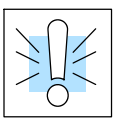
Replace battery with part #D3-D4-BAT

To replace the CPU battery:

- Pull the battery out from the battery retaining clip.
- Lift the clip on the two wire battery connector.
- Slide the battery connector apart.

To install the CPU battery:

- Join the (keyed) battery connector so the red wires match.
- Push gently till the connector snaps closed
- Slide the battery into the battery retaining clip till the battery lies flush in the opening.
- Note the date the battery was changed.



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

CMOS RAM Memory Cartridge Battery Replacement

The CMOS RAM Memory Cartridge battery is used to maintain the contents of the Memory Cartridge RAM when the power is turned off. The life expectancy of this battery is three years.

NOTE: For added security you will want to save the contents of the memory cartridge to either another memory cartridge, cassette tape, or computer disk,to avoid losing the contents of the memory cartridge when the battery is removed. The memory cartridge does however have a built in capacitor to retain the memory for several minutes while the battery is being replaced. If the system has been powered off, you should power-up the CPU with the memory cartridge installed for at least 5 seconds prior to changing the battery. This is done to ensure the capacitor used to maintain the proper voltage levels necessary to retain memory is fully charged.

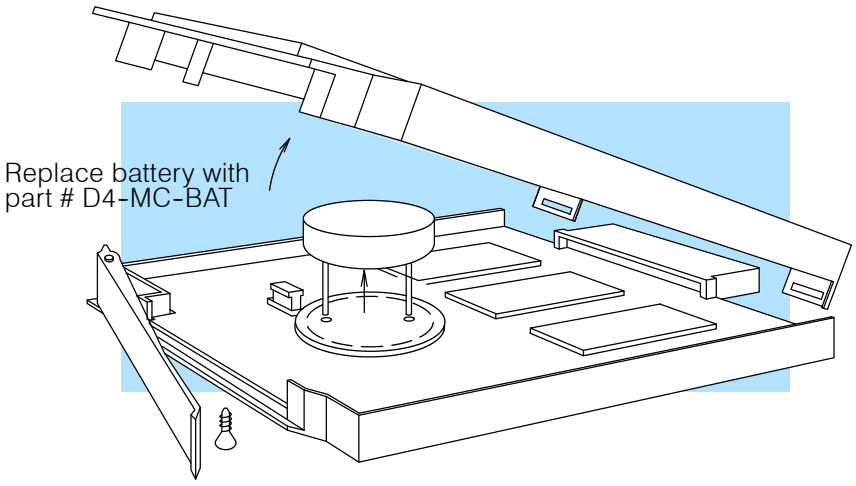
WARNING: Never remove the Memory Cartridge from the CPU when the CPU is powered-up. If the memory cartridge is removed while under power the memory cartridge may become unreliable. Do not attempt to recharge the battery or dispose of it by fire. The battery may explode or release hazardous materials.

To remove the CMOS RAM Memory Cartridge battery:

1. Power down the CPU.
2. Remove the memory cartridge from the CPU by gently pulling the lever down till the cartridge is loosened, then pull the lever straight out .
3. Remove the cover retaining screw.
4. Lift top cover off (lifting at a slight tilt to avoid breaking the guides beside the connector).
5. Grasp battery and lift straight up.

To install the CMOS RAM Memory Cartridge battery:

1. Align the battery leads to the (keyed) holes in the circuit board.
2. Press the battery flush with the circuit board.
3. Place cover back on module (using a slight tilt to catch the guides beside the connector).
4. Install the cover retaining screw.
5. Note the date the battery was changed.



Diagnostics

Diagnostics

DL405 performs over 90 pre-defined diagnostic routines with every CPU scan. The diagnostics have been designed to detect failures for the CPU local base and local expansion bases. These detected errors cover the following: CPUs, expansion units, I/O, bases, communication modules, memory cartridges and batteries. There are two primary error classes: fatal and non-fatal.

Fatal Errors

Fatal errors are errors the CPU has detected that offer a risk of the system not functioning safely or properly. They will either cause the CPU to change from RUN mode to PROGRAM or will disallow changing from PROGRAM mode to RUN mode. If the fatal error is detected while in PROGRAM mode the CPU will not enter the RUN mode until the error has been corrected.

Examples of fatal errors:

- Power supply failure on the CPU base
- Parity error or CPU malfunction
- Certain programming errors

Non-fatal Errors

Non-fatal errors are errors that are flagged by the CPU as requiring attention. They can neither cause the CPU to change from RUN mode to PROGRAM nor do they prevent entering RUN mode. There are special relays the application program can use to detect if a non-fatal error has occurred. The application program can then be used to take the system to an orderly shutdown or switch the PLC out of RUN mode if needed.

Example of non-fatal errors are:

- Battery backup battery voltage low
- All I/O module errors
- Certain programming errors

Finding Diagnostic Information

Diagnostic information can be found in several places with varying levels of message detail.

- The CPU automatically logs error codes and any FAULT messages into two separate tables which can be viewed with the Handheld or ***Direct***SOFT.
- The handheld programmer displays error numbers and short descriptions of the error.
- ***Direct***SOFT provides the error number and an error message.
- Appendix B in this manual has a complete list of error messages sorted by error number.

Many of these messages point to supplemental memory locations which can be referenced for additional related information. These memory references are in the form of V-memory and SPs (special relays).

The following two tables name the specific memory locations that correspond to certain types of error messages. The special relay table also includes status indicators which can be used in programming. For a more detailed description of each of these special relays refer to Appendix D.

V-memory 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 (DL450)	V7746
System	10ms calender timer (DL440/450)	V7747
User-defined	Fault message error code	V7751
I/O configuration	Current module ID code	V7752
I/O configuration	Correct module ID code	V7753
I/O configuration	Base number/Slot number	V7754
Fatal	Error code	V7755
Major	Error code	V7756
Minor	Communications Error code	V7757
Module	Base number/Slot number	V7760
Module	Error code	V7762
Grammatical	Address	V7763
Grammatical	Error Code	V7764

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 Appendix D.

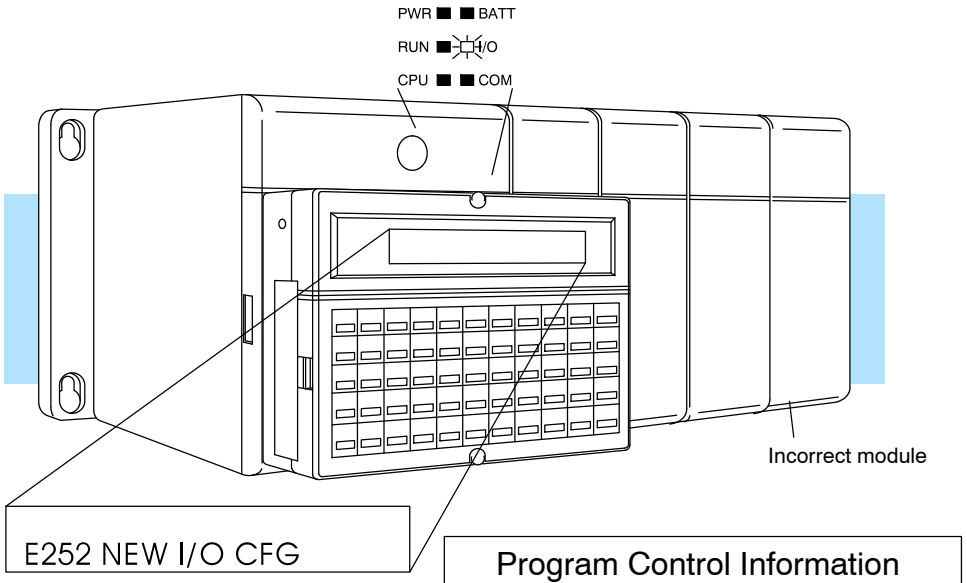
Startup and Real-time Relays		CPU Status Relays		System Monitoring Relays		Accumulator Status Relays		Communication Monitoring Relays	
SP0	First scan relay	SP11	Forced run mode relay	SP40	Critical error relay	SP60	Value less than relay	SP120	Module busy relay Slot 0
SP1	Always ON relay	SP12	Terminal run mode relay	SP41	Warning relay	SP61	Value equal to relay	SP121	Communication error relay Slot 0
SP3	1 minute relay	SP13	Test run mode relay	SP43	Battery low relay	SP62	Value greater than relay	SP122	Module busy relay Slot 1
SP4	1 second relay	SP14	Break relay 1	SP44	Program memory error relay	SP63	Zero relay	SP123	Communication error relay Slot 1
SP5	100 millisecond relay	SP15	Test program mode relay	SP45	I/O error relay	SP64	Half borrow relay	SP124	Module busy relay Slot 2
SP6	50 millisecond relay	SP16	Terminal program mode relay	SP46	Communications error relay	SP65	Borrow relay	SP125	Communication error relay Slot 2
SP7	Alternate scan relay	SP17	Forced stop mode relay	SP47	I/O configuration error relay	SP66	Half carry relay	SP126	Module busy relay Slot 3
		SP21	Break relay 2	SP50	FAULT instruction relay	SP67	Carry relay	SP127	Communication error relay Slot 3
		SP22	Interrupt enabled relay	SP51	Math timeout relay	SP70	Sign relay	SP130	Module busy relay Slot 4
		SP25	CPU battery disabled relay	SP52	Grammatical error relay	SP71	Pointer reference error	SP131	Communication error relay Slot 4
		SP26	I/O update disable relay (DL440)	SP53	Math/Table pointer error	SP73	Overflow relay	SP132	Module busy relay Slot 5
		SP27	Selective I/O update disable relay (DL440)	SP54	Communication error	SP75	Data error relay	SP133	Communication error relay Slot 5
		SP30	Dipswitch 1 status relay	SP56	Table instruction overrun relay	SP76	Load zero relay	SP134	Module busy relay Slot 6
		SP31	Dipswitch 2 status relay					SP135	Communication error relay Slot 6
		SP32	Dipswitch 3 status relay					SP136	Module busy relay Slot 7
		SP33	Dipswitch 4 status relay					SP137	Communication error relay Slot 7

I/O Module Codes

Each system component has a code identifier. This code identifier is used in some of the error messages related to the I/O modules. The following table list these codes.

Code (Hex)	Component Type	Code (Hex)	Component Type
01	CPU	21	8 pt. Input
02	Expansion Unit	28	16 pt. Output, FL series Analog Output
03	I/O Base	2B	16 pt. Input, FL series Analog Input, Interrupt
11	DCM, All CoProcessor Modules	30	32 pt. Output, DL series Analog Output
12	Remote Master, Slice Master	3F	32 pt. Input, DL series Analog Input
18	High Speed Counter, Magnetic Pulse Input	7F	Abnormal
20	8 pt. Output	FF	No module detected

The following diagram shows an example of how the I/O module codes are used:



Program Control Information

V7752	0028	Current module ID code
V7753	0026	Current module ID code
V7754	0002	Location of conflict
V7756	0252	Module Error Code
SP45	⊣⊢	I/O abnormality

Error Message Tables

X	✓	✓
430	440	450

The DL440 and DL450 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 11 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.

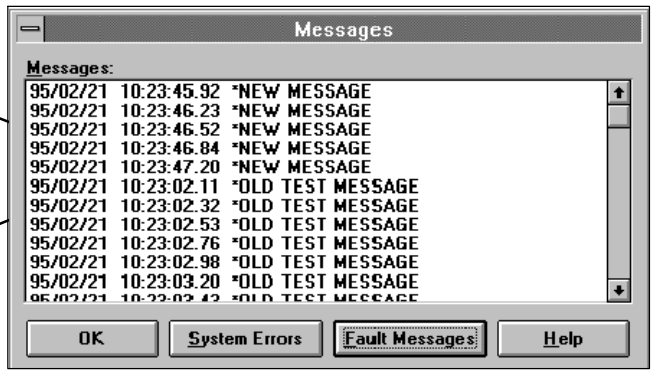
When an error or message is triggered, it is put into the first available table location. *Therefore, the most recent error message may not appear in the first row of the table.* If the table is full when an error occurs, the oldest error is pushed (erased) from the table and the new error is inserted in the row.

The following diagram shows an example of a the Fault Message table as shown in ***Direct***SOFT. You can access the error code table and the message table through ***Direct***SOFT's PLC Diagnostic sub-menus. Details on how to access these logs are provided in the ***Direct***SOFT 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.



You cannot view the entire table at one time with the handheld programmer. Instead, the messages automatically appear on the handheld programmer display as they occur. The message will remain on the display as long as the Fault instruction is being executed. The following examples show you how to use the Handheld and AUX Function 5C to show the error codes. The most recent error or message is always displayed. You can use the PREV and NXT keys to scroll through the messages.

Use AUX 5C to view the tables

CLR AUX 5 SHFT C ENT

```
AUX 5C SHOW ERR/MSG
      ERR OR MSG
```

Arrow key selects Errors or Messages

SHFT → ENT

```
AUX 5C SHOW ERR/MSG
      ERR OR MSG
```

Example of a message display

```
PUMP 3 FAILED
04/22/93 17:30:00
```

System Error Codes

X	✓	✓
430	440	450

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 DL405 systems generate.

The following errors are captured in the System error log upon first detection or reoccurrence.

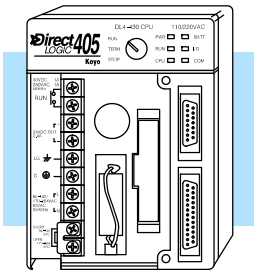
Error Code	Description	Error Code	Description
E003	Software time-out	E155	RAM failure
E004	Invalid instruction (RAM parity error in the CPU)	E201	Terminal block missing
E041	CPU battery low	E202	Missing I/O module
E043	Memory cartridge battery low	E203	Blown fuse
E099	Program memory exceeded	E206	User 24V power supply failure
E101	CPU memory cartridge missing	E250	Communication failure in the I/O chain
E104	Write fail	E251	I/O parity error
E151	Invalid Command	E252	New I/O configuration

These errors are captured in the System error log if they exist when the CPU attempts to transition to RUN mode.

Error Code	Description	Error Code	Description
E401	Missing END statement	E431	Invalid ISG/SG address
E402	Missing LBL	E432	Invalid jump (GOTO) address
E403	Missing RET	E433	Invalid SBR address
E404	Missing FOR	E434	Invalid RTC address
E405	Missing NEXT	E435	Invalid RT address
E406	Missing IRT	E436	Invalid INT address
E412	SBR/LBL > 64	E437	Invalid IRTC address
E413	FOR/NEXT > 64	E438	Invalid IRT address
E421	Duplicate stage reference	E440	Invalid Data Address
E422	Duplicate SBR/LBL reference	E441	ACON/NCON
E423	Nested loops		

CPU Status Indicators

The DL405 CPUs have indicators on the front to help you diagnose problems with the system. The table below gives a quick reference of potential problems associated with each status indicator. Following the table is a detailed description of each indicator.



Indicator	Status	Potential Causes
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 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	ON, or flashing	1. Flashing at 2 Hz: CPU battery low 2. Flashing at 0.5 Hz: Memory cartridge battery low (DL440/DL450 only) 3. On constantly: Both the CPU and MC batteries are low 4. CPU or Memory cartridge battery missing, or disconnected
DIAG (DL450)	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
COM (DL430/ DL440)	ON	1. Device port setup incorrect 2. Cabling error 3. Grounding problem 4. Electrical noise 5. Device port faulty
TXD (DL450)	OFF	1. The CPU is not transmitting data on the secondary ports (ports 1, 2, and 3), due to programming error 2. The CPU is not in Run mode
RXD (DL450)	OFF	1. External device is not transmitting to CPU secondary ports (ports 1, 2, and 3) 2. Communications cable is defective, or not connected

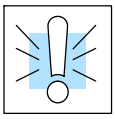
PWR Indicator

✓ ✓ ✓
430 440 450

In general there are four reasons for the CPU or expansion unit power status LED (PWR) to be OFF:

1. Power to the CPU is an improper voltage, or the power range jumper (110/220 select) does not match the power input voltage.
2. CPU power supply is faulty.
3. Other component(s) have the power supply shut down. An I/O module in the base which has shorted is a possibility.
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.

1. First, disconnect the external power.
2. Verify that all external circuit breakers or fuses are still intact.
3. Check all incoming wiring for loose connections. If you're using a separate termination block, check those connections for accuracy and integrity.
4. If the connections are acceptable, reconnect the system power and verify the voltage at the CPU power input is within specification. If the voltage is not correct, shut down the system and correct the problem.
5. 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 or DC 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, System Design and Configuration.

Run Indicator

430 440 450

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, in which case the CPU LED should be on.

- If you are attempting to enter the run mode by a programming device and the CPU will not enter the run mode, first make sure the key switch is in the TERM position and then try to enter the run mode.
- If you are using the keyswitch to change to RUN mode and the CPU does not respond, attach a programming device to diagnose what error is being returned.
- If the indicator is flashing, the CPU is in firmware upgrade mode.

Both of the programming devices, handheld programmer and ***Direct***SOFT, will return a error message and depending on the error may also recommend an AUX function to run that will aid in further diagnosing the problem. The most common programming error is "Missing END Statement". All application programs require an END statement for proper termination. Appendix B lists all the error codes.

CPU Indicator

✓	✓	✓
430	440	450

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, the system should be closely monitored and every effort should be made to try to determine the cause of the problem. You will find this problem is sometimes caused by high frequency electrical noise introduced into the CPU from a 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, replace the CPU.

BATT Indicator

✓	✓	✓
430	440	450

If the BATT indicator is on or flashing, either or both the CPU/Memory Cartridge batteries are low (2.5V or less). The DL430 does not have a memory cartridge. The battery voltage is continuously monitored while the system voltage is being supplied.

BATT LED Status	Error Condition
Flashing at 2Hz	CPU battery is low
Flashing at 0.5 Hz	Memory Cartridge battery is low
ON constantly	Both the CPU and Memory Cartridge batteries are low

Procedures for how to replace either of the batteries can be found in the Hardware Maintenance section earlier in this chapter.

DIAG Indicator

X	X	✓
430	440	450

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

✓	✓	✓
430	440	450

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

I/O error detection for remote I/O will be covered in the DL405 Remote and Slice I/O manual.

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:

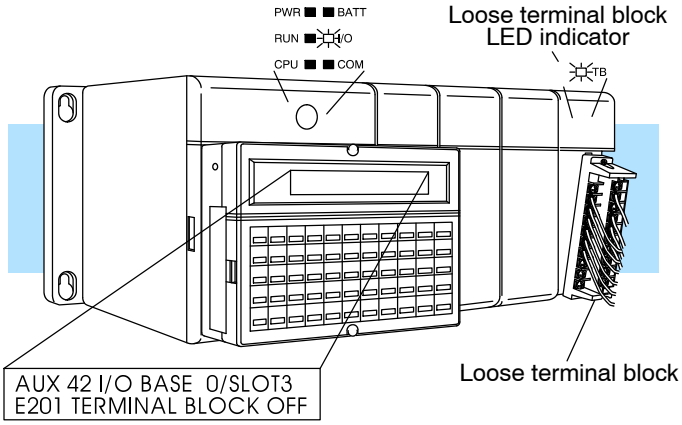
Indicator	Error condition
TB	Loose or missing terminal block
24V	External 24V power supply not providing the correct voltage
FU	Module fuse has blown, (check the I/O modules specification sheets to see if the fuse is replaceable)

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 AUX 42 from the handheld programmer or I/O diagnostics in ***Direct***SOFT. Both options will provide the base number, the slot number and the problem with the module. Once the problem is corrected the indicators will reset.

An I/O error will not cause the CPU to switch from the run to program mode, however there are special relays (SPs) available in the CPU which will allow this error to be read in ladder logic. The application program can then take the required action such as entering the program mode or initiating an orderly shutdown. The following figure shows a example of the failure indicators.

Internal diagnostics		
V7756	0201	Major Error Code
V7760	0003	Error Location (Base 00, Slot 03)
V7762	0201	Module Error Code
SP45	⊣⊢	I/O Error



COM Indicator

430	440	450
✓	✓	X

The COM indicator on DL430 and DL440 CPU turns ON when the CPU has detected a communication error on one of the two communication ports built into the CPU. The most common causes for this error are:

- 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 to the port is sending data incorrectly.
- A grounding difference exists between the two devices.
- Electrical noise is causing intermittent errors
- The CPU has a bad communication port and the CPU should be replaced

If an error occurs the indicator will come on and stay on until a successful communication has been completed. If the cable and its connections are OK, try doing a power cycle on the devices at both ends of the communications cable.

TXD and RXD Indicators

430	440	450
X	X	✓

The TXD and RXD indications on the DL450 CPU work like the identically-named LED indicators found on modem devices. The TXD and/or RXD indicators 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.

The TXD and RXD indicators turn on when data is transmitted on any of the four ports on the DL450. Therefore, when ***Direct***SOFT, or a HPP, 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.

I/O Module 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 counting, specialized communications, and so on
- A blown fuse in your machine or panel (the DL105 does not have internal I/O fuses)
- A loose terminal block
- The auxiliary 24 VDC supply has failed
- An Input or Output point has failed

Some Quick Steps

When troubleshooting the DL series I/O modules there are a few facts you should be aware of. These facts may assist you in quickly correcting an I/O problem.

- The output modules cannot detect shorted or open output points. If you suspect one or more points on a output module to be faulty, measure the voltage drop from the common to the suspect point. Remember when using a Digital Volt Meter, leakage current from an output device such as a triac or a transistor must be considered. A point which is off may appear to be on if no load is connected to the point.
- The I/O point status indicators on the modules are logic side indicators. This means the LED which indicates the on or off status reflects the status of the point with respect to the CPU. On a output module the status indicators could be operating normally while the actual output device (transistor, triac etc.) could be damaged. With an input module if the indicator LED is on, the input circuitry should be operating properly. Verify the LED goes off when the input signal is removed.
- Leakage current can be a problem when connecting field devices to I/O modules. False input signals can be generated when the leakage current of an output device is great enough to turn on the connected input device. To correct this install a resistor in parallel with the input or output of the circuit. The value of this resistor will depend on the amount of leakage current and the voltage applied but usually a 10K to 20K ohm resistor will work. Verify the wattage rating of the resistor is correct for your application.
- The fuse blown indicator on a 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 a load applied to the output to create the voltage drop before it can be reported by the module.
- The easiest method to determine if a module has failed is to replace it if you have a spare. However, if you suspect another device to have caused the failure in the module, that device may cause the same failure in the replacement module as well. As a point of caution, you may want to check devices or power supplies connected to the failed module before replacing it with a spare module.

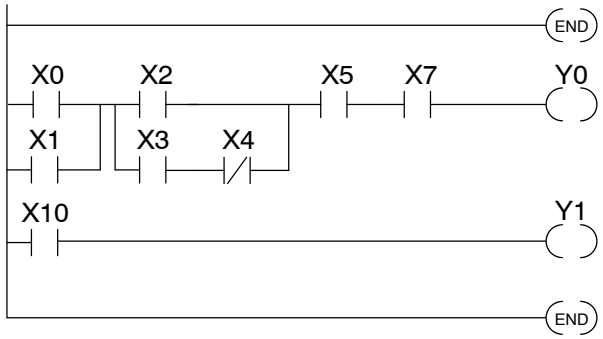
Testing Output Points

Output points can be set on or off in the DL405 series CPUs but they cannot be forced in such a way to override ladder logic. 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 keyswitch to TERM.
2	Use a handheld programmer or ***Direct***SOFT 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 running and turning I/O points on or off).
6	Change to Run mode using the handheld programmer or ***Direct***SOFT. We must do this to enable the outputs.
7	Use the programming device 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.

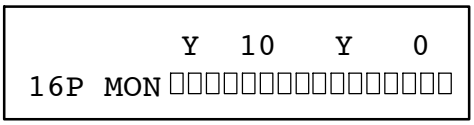
WARNING: Depending on your application, forcing I/O points may cause unpredictable machine operation that can result in a risk of personal injury or equipment damage. Make sure you have taken all appropriate safety precautions prior to testing any I/O points.

Handheld Programmer Keystrokes Used to Test an Output Point



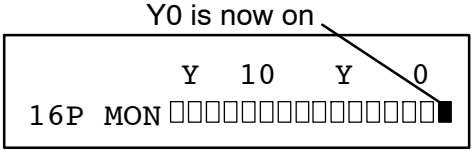
From a clear display, use the following keystrokes

Y(OUT) 0 BIT ST



Turn the output on (or off)

Y(OUT) 0 SHFT ON



Noise Troubleshooting

Electrical Noise Problems

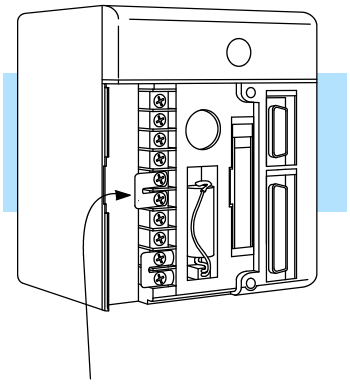
Noise is one of the most difficult problems to diagnose. Electrical noise can enter a system in many ways, which are divided into two categories, conducted or radiated:

- Conducted noise is when the electrical interference is introduced into the system by way of an attached wire, panel connection, etc. It may enter through an I/O module, a power supply connection, the communication ground connection, or the chassis ground connection.
- Radiated noise is electrical interference introduced into the system through the air as radio waves, without a direct electrical connection.

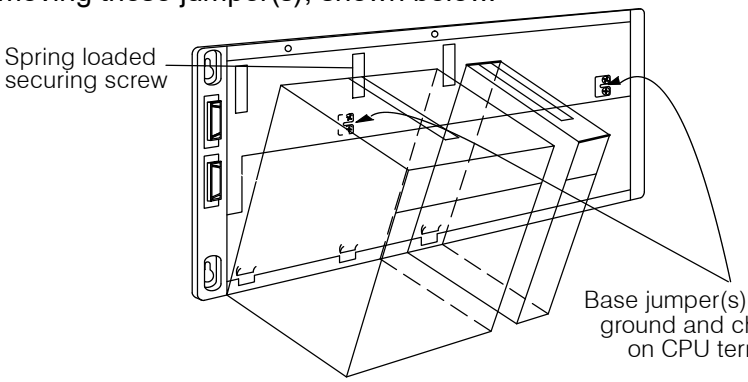
Reducing Electrical Noise

The following tips can help reduce electrical noise enough for normal operation.

- Most noise problems result from improper grounding of the system. A good earth ground can be the single most effective way to correct noise problems. If a ground is not available, install a ground rod as close to the system as possible. Ensure all ground wires are single point grounds and are not daisy chained from one device to another. Ground metal enclosures around the system. A loose wire is no more than a large antenna waiting to introduce noise into the system; therefore, you should tighten all connections in your system. Loose ground wires are more susceptible to noise than the other wires in your system. Review Chapter 2 Installation and Safety Guidelines if you have questions regarding how to ground your system.
- Electrical noise can enter the system through the power source for the CPU and I/O. Installing an isolation transformer for all AC sources can correct this problem. DC sources should be well grounded, good quality supplies. Switching DC power supplies commonly generate more noise than linear supplies.
- Separate input wiring from output wiring. Never run analog I/O wiring or low voltage discrete I/O wiring close 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 shown to the left.
- In rare instances you may want to isolate logic ground from chassis ground. There is a single jumper on the 4 slot base and two jumpers on the 6 and 8 slot bases for this purpose. Isolation can be obtained by removing these jumper(s), shown below.



Optional CPU jumper connects logic ground and chassis ground



Machine Startup and Program Troubleshooting

The DL405 CPUs 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
- Test Modes
- Run Time Edits
- Special Instructions

Syntax Check

Even though the Handheld Programmer and ***Direct***SOFT provide error checking during program entry, you may want to check a program that has been modified. Both programming devices offer a way to check the program syntax. For example, you can use AUX 21, CHECK PROGRAM to check the program syntax from a Handheld Programmer, or you can use the PLC Diagnostics menu option within ***Direct***SOFT. This check will find a wide variety of programming errors. The following example shows how to use the syntax check with a Handheld Programmer.

Use AUX 21 to perform syntax check

CLR AUX 2 1 ENT ENT

```
AUX 21 CHECK PROGRAM
1:SYN 2:DUP REF
```

Select syntax check

1 ENT (This may take a minute or so.)

```
BUSY
```

One of two displays will appear

Error Display (example)

```
$ 8 E401 MISSING END
TMRA   T 002   K00050
```

(shows location in question)

Syntax OK display

```
NO SYNTAX ERROR
?
```

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 NO SYNTAX ERROR message appears.

Duplicate Reference Check

You can also check for multiple uses of the same output coil. Both programming devices offer a way to check for this condition. For example, you can AUX 21, CHECK PROGRAM to check for duplicate references from a Handheld Programmer, or you can use the PLC Diagnostics menu option within ***DirectSOFT***. The following example shows how to perform the duplicate reference check with a Handheld Programmer.

Use AUX 21 to perform syntax check

AUX 2 1 ENT ENT

```
AUX 21 CHECK PROGRAM
1:SYN 2:DUP REF
```

Select Duplicate Reference check

2 ENT (This may take a minute or so.)

```
BUSY
```

One of two displays will appear

Error Display (example)

```
$ 12 E471 DUP COIL REF
OUT    Y 0000
```

(shows location in question)

Syntax OK display

```
NO DUP REFS
?
```

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.

TEST-PGM and TEST-RUN Modes

Test Mode allows the CPU to start in TEST-PGM mode, enter TEST-RUN mode, run a fixed number of scans, and then return to TEST-PGM mode. You can select from 1 to 65,535 scans. You can select this operation from either the Handheld Programmer (AUX 12) or ***Direct***SOFT via a PLC Modes menu option.

With the Handheld, the actual mode entered when you first select Test Mode depends on the mode of operation at the time you make the request. If the CPU is in RUN mode, then TEST-RUN is entered. If the mode is PGM, then TEST-PGM is entered. ***Direct***SOFT provides more flexibility in selecting the various modes with different menu options. The following example shows how you can use the Handheld to select the Test Modes

Use AUX 12 to enter Test Mode

AUX 1 2 ENT ENT

```
MODE = TEST-PGM
```

To specify the number of scans. . .

CLR 1 SHFT TEST

(CPU runs scans and returns to Test-PGM)

```
NO. OF SCANS?
```

To switch to TEST-PGM mode . . .

CLR 2 SHFT TEST

ENT (to confirm the return to TEST-PGM)

```
STOP SCAN?
```

To switch to TEST-RUN mode . . .

CLR 3 SHFT TEST

ENT (to confirm the entry of TEST-RUN)

```
START SCAN?
```

With the Handheld Programmer you gain some advantages by using Test Mode.

- The Handheld Programmer status displays are more detailed.
- You can enable the CPU to hold output states.

Test Mode Displays: For some instructions, the TEST-RUN mode display is more detailed than the status displays shown in RUN mode. The following diagram shows an example of a Timer instruction display during TEST-RUN mode.

RUN Mode

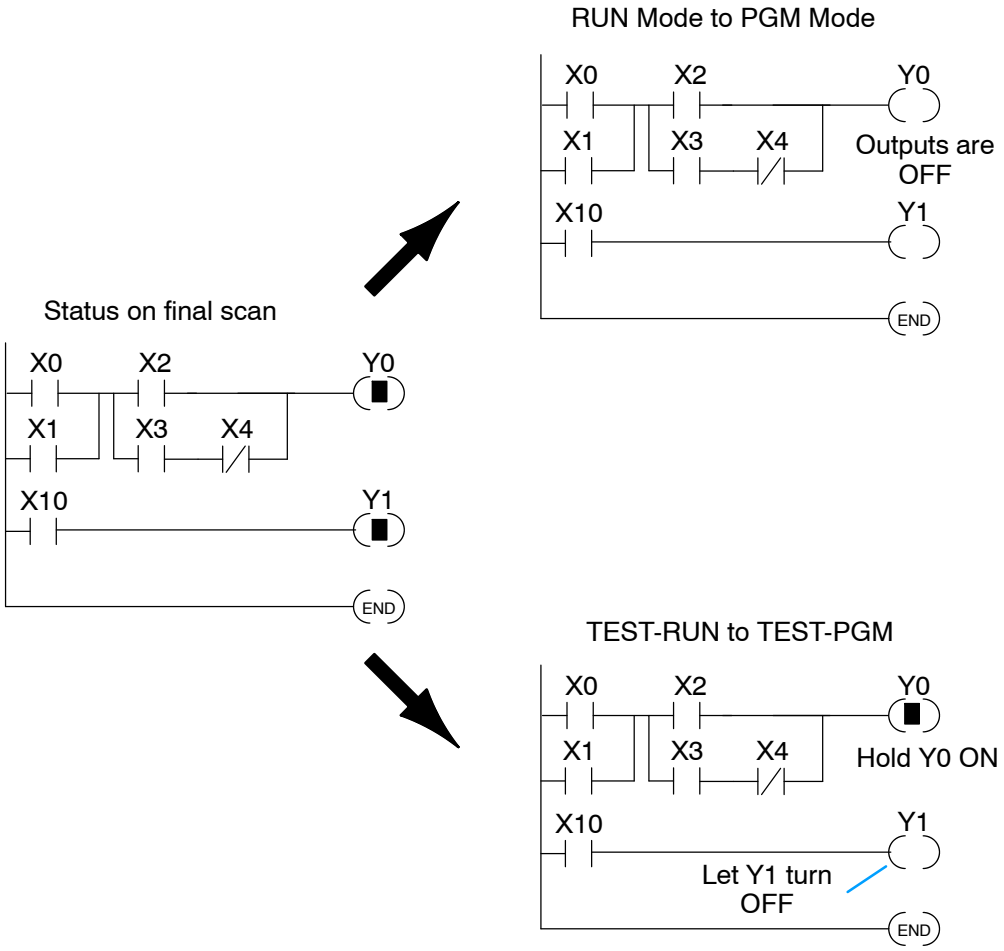
```
$    3                  ■
TMR    T 000   K0500
```

TEST-RUN Mode

```
$    3   T=510          ■
TMR    T 000   K0500
```

Current Value

Holding Output States: In normal RUN mode, the outputs are turned off when you return to PGM mode. In TEST-RUN mode you can set each individual output to either turn off or hold its last output state on the transition to TEST-PGM mode. The ability to hold the output states is especially useful, since It allows you to maintain key system I/O points for examination. The following diagram shows the differences between RUN and TEST-RUN modes.

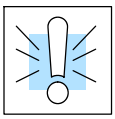


You can use AUX 58 on the Handheld Programmer to select the action for each individual output.

Run Time Edits

X	✓	✓
430	440	450

The DL440 and DL450 CPUs allow you to make changes to the application program during Run Mode. These edits are not "bumpless." Instead, CPU scan is momentarily interrupted (and the outputs are maintained in their current state) until the program change is complete. This means if the output is off, it will remain off until the program change is complete. If the output was on, it will remain on.



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

1. **If there is a syntax error in the new instruction, the CPU *will not* enter the Run Mode.**
2. **If you delete an output coil reference and the output was on at the time, the output will remain on until it is forced off with a programming device.**
3. **Input point changes are not acknowledged during Run Time Edits. So, if you're using a high-speed operation and a critical input comes on, the CPU may not see the change.**

You can use either the Handheld Programmer or ***Direct***SOFT to edit the program during Run Mode. The following pages show a brief example of how to do this with the Handheld Programmer. You use AUX 14 to edit the program during Run Mode. We've already shown you how to select the various AUX functions, but a few things are different with AUX 14.

- Once you select AUX 14 the Handheld RUN LED starts blinking. This indicates the a Run Mode edit is in progress.
- If you had displayed an address just before selecting AUX 14, that address will automatically appear. So, you can search for an address or instruction *before* you select AUX 14 or *after* you select AUX 14.

Select AUX 14, Run Time Edit

AUX 1 4 ENT

```
AUX 1* OPERATING MODE
AUX 14 RUN TIME EDIT
```

Press ENT to select AUX 14 and display the address

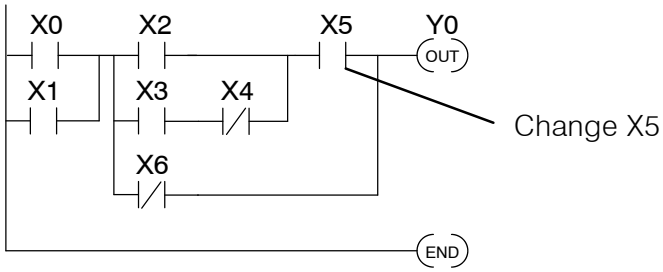
ENT

```
$xxxxx
STR    X 0001
```

Changing an Instruction During Run Mode

Once you've found the instruction you can change it very easily. The following example shows you how to change the X5 contact to X10.

Ladder Representation



Identify the Instruction

ADDRESS	INSTRUCTION	DESCRIPTION
0	STR X0	Starts branch 1 with X0
1	OR X1	Joins X1 in parallel with X0
—	—	—
—	—	—
6	AND X5	Starts branch 4 with X5
—	—	—
—	—	—
10	END	Ends the program

FIND the Address

AND X(IN) 5 FIND

```
            SEARCHING
AND X5
```

```
$     6
AND     X 0005
```

Change the Instruction

AND X(IN) 10 ENT

```
$     6
WANT TO ALTER?
```

Press CLR to abort the edit or ENT to accept

ENT

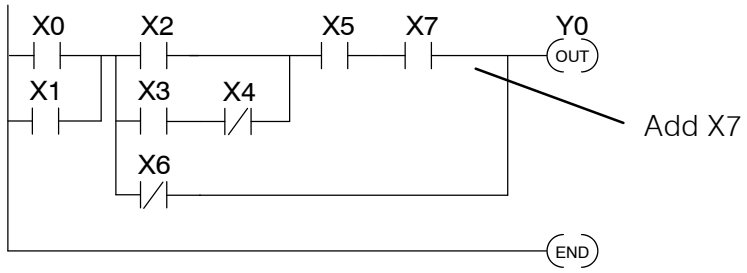
(If you press ENT, the change is accepted and the next address is displayed. If you pressed CLR, the current address is displayed.

```
$     7
ORN     X 0006
```

Inserting an Instruction During Run Mode

Inserting an instruction during Run Mode works almost exactly the same as it does during Program Mode. Remember, INSERT adds an instruction *before* the instruction that is being displayed and the remaining addresses increment.

Ladder Representation



Identify the Instruction

ADDRESS	INSTRUCTION	DESCRIPTION
0	STR X0	Starts branch 1 with X0
1	OR X1	Joins X1 in parallel with X0
—	—	—
6	AND X5	Starts branch 4 with X5
	AND X7	Adds X7 in series with X5
7	ORN X6	Joins X6 (NOT) in parallel
—	—	—
10	END	Ends the program

Insert before

FIND the Address

OR NOT X(IN) 6 FIND

```
            SEARCHING
ORN X6
```

```
$     7
ORN     X 0006
```

Insert the New Instruction

AND X(IN) 7 SHFT INS

```
$     7
WANT TO INSERT?
```

Press CLR to abort the edit or ENT to accept

ENT

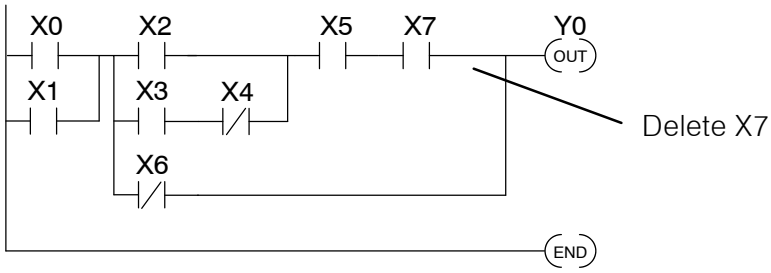
(If you press ENT, the change is accepted and the next address is displayed. If you pressed CLR, the current address is displayed.

```
$     8
ORN     X 0006
```

Deleting an Instruction During Run Mode

Deleting an instruction during Run Mode works almost exactly the same as it does during Program Mode. Remember, this operation deletes the instruction that is currently being displayed and the remaining addresses decrement.

Ladder Representation



Identify the Instruction

	ADDRESS	INSTRUCTION	DESCRIPTION
	0	STR X0	Starts branch 1 with X0
	1	OR X1	Joins X1 in parallel with X0
	—	—	—
Delete	6	AND X5	Starts branch 4 with X5
	7	AND X7	Adds X7 in series with X5
	—	—	—
	11	END	Ends the program

FIND the Address

AND X(IN) 7 FIND

```
          SEARCHING
AND X7
```

```
$     7
AND     X 0007
```

Delete the Instruction

SHFT DEL

```
$     7
WANT TO DELETE?
```

Press CLR to abort the edit or ENT to accept

ENT

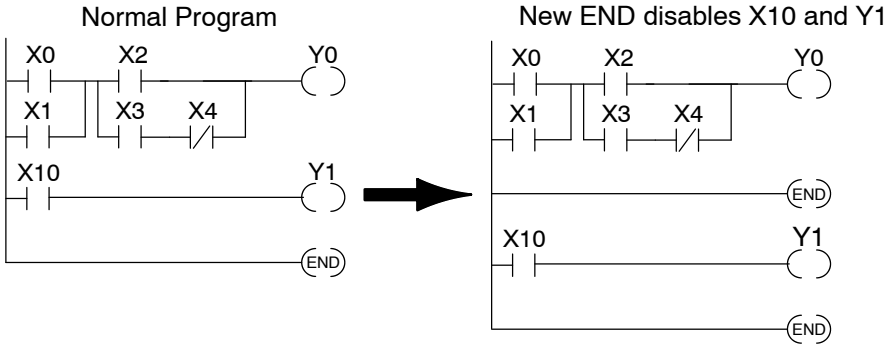
(If you press ENT, the change is accepted and the next address is displayed. If you pressed CLR, the current address is displayed.

```
$     7
ORN     X 0006
```

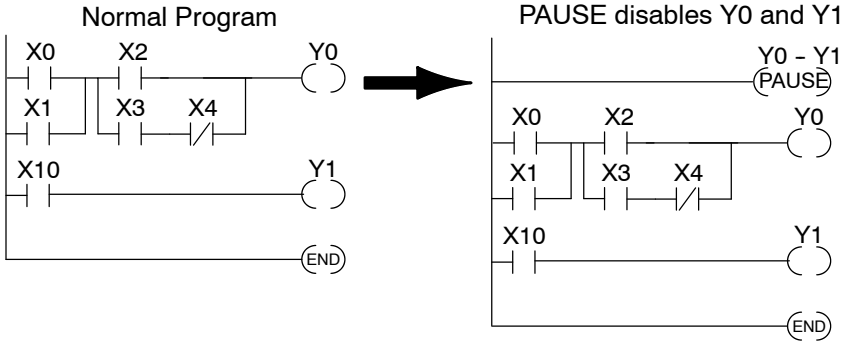
Special Debug Instructions

There are several instructions that can be used to help you debug your program during machine startup operations: END, PAUSE, STOP, and BREAK.

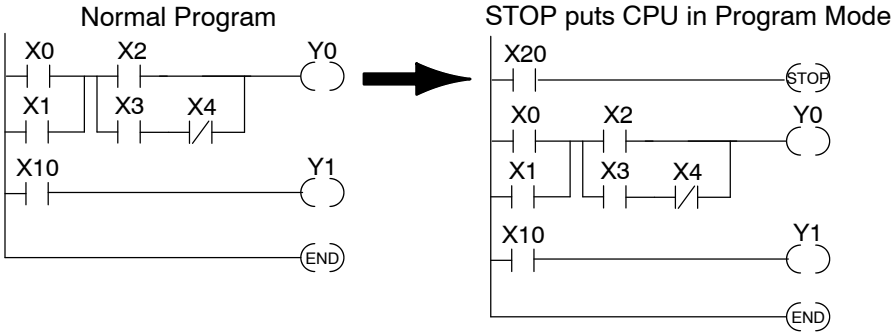
END Instruction: You can quickly disable part of the program by inserting an END statement prior to the portion that should be disabled. The CPU assumes that is the end of the program. The following diagram shows an example.



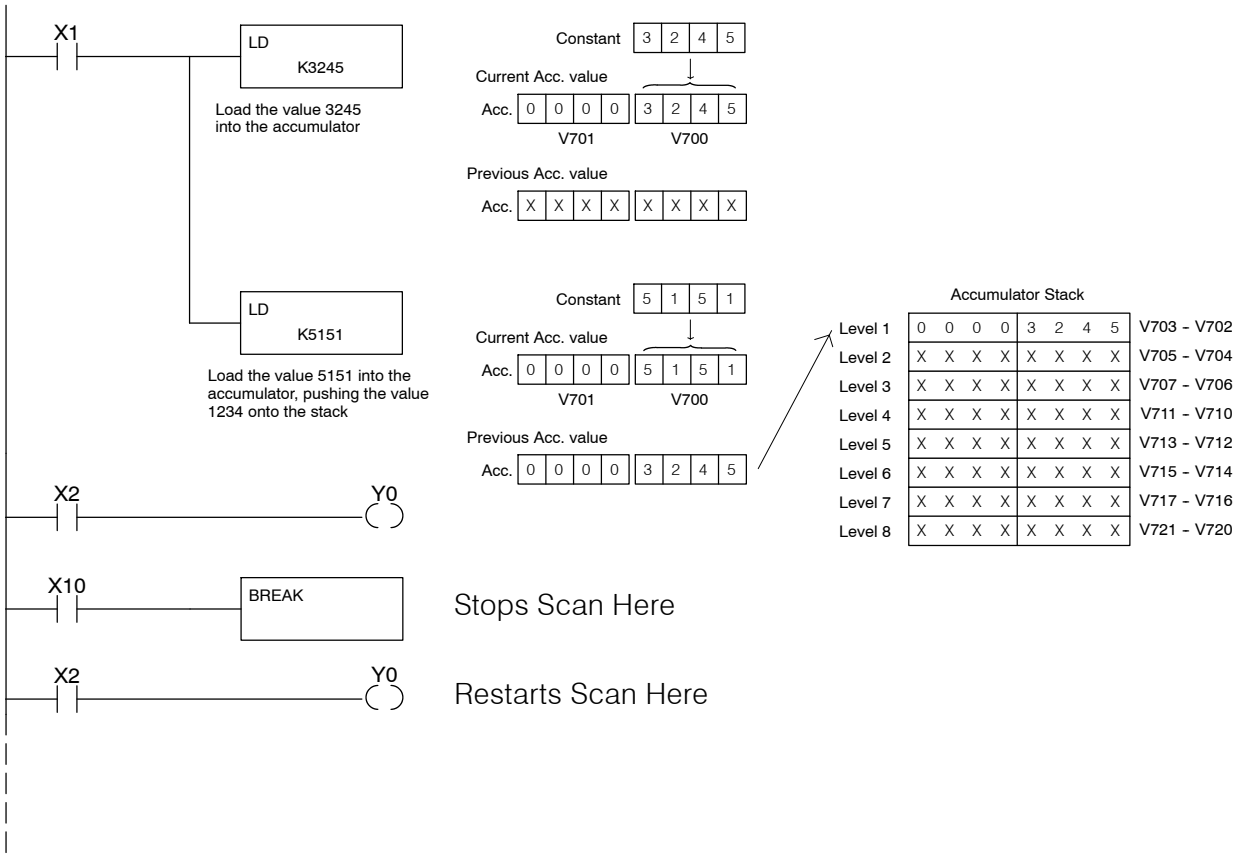
PAUSE Instruction: This instruction provides a quick way to allow the inputs (or other logic) to operate while disabling selected outputs. The output image register is still updated, but the output status is not written to the modules. For example, you could make this conditional by adding an input contact or CR to control the instruction with a switch or a programming device. Or, you could just add the instruction without any conditions so the selected outputs are always disabled.



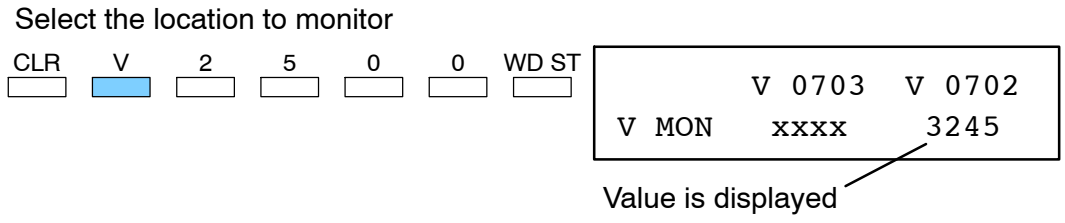
STOP Instruction: Sometimes during machine startup you need a quick way to turn off all the outputs and return to Program Mode. In addition to the Test Modes, you can also use the STOP instruction. When executed, a STOP causes the CPU to exit Run Mode and enter Program Mode. The following program does this. Remember, all outputs are turned off during Program Mode.



BREAK Instruction: If you have a DL440 CPU you can also use the BREAK instruction to stop the program scan. As long as the BREAK instruction is active the scan is stopped and the CPU enters a special mode, TEST-HALT. You have to use either the Handheld Programmer or ***Direct***SOFT to restart the scan by placing the CPU back in Run Mode. When the CPU returns to Run Mode the scan resumes at the point of the break. For example, if you are using several data instructions it may be helpful to stop the scan and examine the accumulator and accumulator stack. The following diagram shows an example.



In the example, input X10 triggers the BREAK instruction. The CPU will stop scanning the program at this point. Now you could easily see how the program instructions have affected the accumulator or accumulator stack. The following diagram shows how you could use the Handheld to examine the accumulator stack.



When you use the Handheld to return to Run Mode, the CPU starts scanning the program at the rung following the BREAK.

See the Errata Sheet at the beginning of this file. We added a procedure named "Reset the PLC to Factory Defaults".

Auxiliary Functions

A

In This Appendix. . . .

— Introduction
— AUX 1* — Operating Modes
— AUX 2* — RLL Operations
— AUX 3* — V-memory Operations
— AUX 4* — I/O Configuration
— AUX 5* — CPU Configuration
— AUX 6* — Handheld Programmer Configuration
— AUX 7* — Memory Cartridge Operations
— AUX 8* — Password Operations

Introduction

What are Auxiliary Functions?

Many CPU setup tasks involve the use of Auxiliary (AUX) Functions. The AUX Functions perform various operations, ranging from simple operating mode changes to copying programs to memory cartridges. They are divided into categories that affect different system resources. You can access AUX Functions from ***Direct***SOFT or from the Handheld Programmer. Some AUX Functions are designed specifically for the Handheld Programmer setup, so they will not be available with the ***Direct***SOFT package. You may need to supplement this Appendix with information from the documentation for your choice of programming device.

AUX Function and Description		430	440	450	HPP
AUX 1* — Operating Mode					
11	Go to Run Mode	✓	✓	✓	×
12	Go to Test Mode	✓	✓	✓	×
13	Go to Program Mode	✓	✓	✓	×
14	Run Time Edit	×	✓	✓	×
AUX 2* — RLL Operations					
21	Check Program	✓	✓	✓	×
22	Change Reference	×	✓	✓	×
23	Clear Ladder Range	✓	✓	✓	×
24	Clear Ladders	✓	✓	✓	×
25	Select MC or Flash Memory	×	×	✓	×
26	Copy MC Contents to Flash	×	×	✓	×
27	Copy Flash contents to MC	×	×	✓	×
28	Verify Flash contents = MC	×	×	✓	×
AUX 3* — V-Memory Operations					
31	Clear V Memory	✓	✓	✓	×
32	Clear V Range	✓	✓	✓	×
33	Find V-memory Value	×	✓	✓	×
AUX 4* — I/O Configuration					
41	Show I/O Configuration	✓	✓	✓	×
42	I/O Diagnostics	✓	✓	✓	×
44	Power up I/O Configuration Check	✓	✓	✓	×
45	Select Configuration	✓	✓	✓	×
46	Configure I/O	×	✓	✓	×
47	Intelligent I/O	✓	✓	✓	×

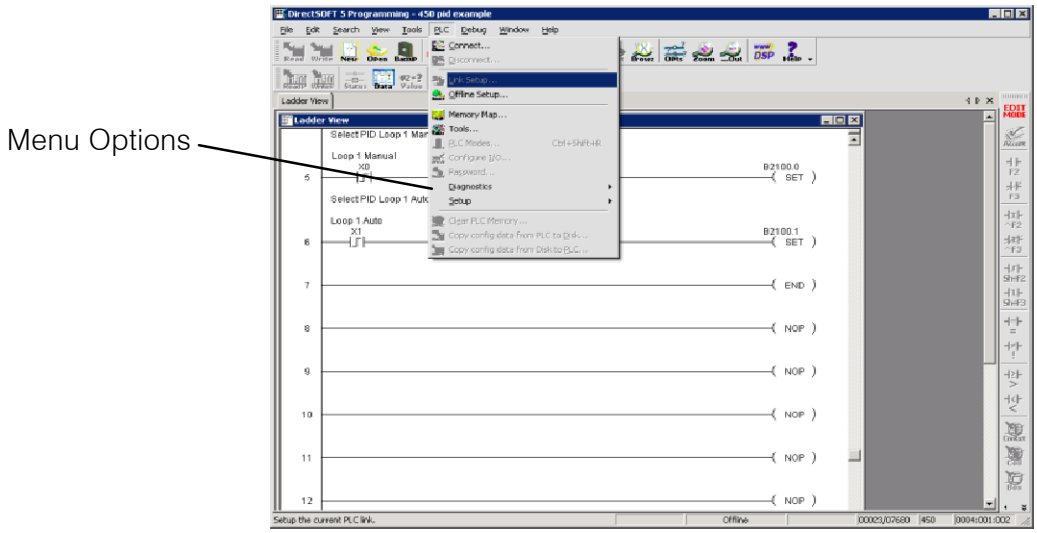
✓ — supported

× — not supported or not applicable

AUX Function and Description		430	440	450	HPP
AUX 5* — CPU Configuration					
51	Modify Program Name	✓	✓	✓	×
52	Display / Change Calendar	×	✓	✓	×
53	Display Scan Time	✓	✓	✓	×
54	Initialize Scratchpad	✓	✓	✓	×
55	Set Watchdog Timer	✓	✓	✓	×
56	Configure Comm. Ports	✓	✓	✓	×
57	Set Retentive Ranges	✓	✓	✓	×
58	Test Operations	✓	✓	✓	×
5C	Display Error History	×	✓	✓	×
5D	Select PLC Scan Mode	×	×	✓	×
AUX 6* — Handheld Programmer Configuration					
61	Show Revision Numbers	✓	✓	✓	×
62	Beeper On / Off	×	×	×	✓
63	Backlight On / Off	×	×	×	✓
64	Select Online / Offline	×	×	×	✓
65	Run Self Diagnostics	×	×	×	✓
AUX 7* — Memory Cartridge Operations					
71	CPU to Memory Cartridge	✓	✓	✓	✓
72	Memory Cartridge to CPU	✓	✓	✓	✓
73	Compare MC with CPU	✓	✓	✓	✓
74	Memory Cart. Blank Check	×	×	×	✓
75	Clear Memory Cartridge	×	×	×	✓
76	Display Memory Cart. Type	✓	✓	✓	✓
77	Tape to Memory Cartridge	×	×	×	✓
78	Memory Cartridge to Tape	×	×	×	✓
79	Compare MC with Tape	×	×	×	✓
AUX 8* — Password Operations					
81	Modify Password	×	✓	✓	×
82	Unlock CPU	×	✓	✓	×
83	Lock CPU	×	✓	✓	×

Accessing AUX Functions via *Direct*SOFT

DirectSOFT provides various menu options during both online and offline programming. Some of the AUX functions are only available during online programming, some only during offline programming, and some during both online and offline programming. The following diagram shows an example of the PLC operations menu available within ***Direct***SOFT.



Accessing AUX Functions via the Handheld Programmer

You can also access the AUX functions by using a Handheld Programmer. Plus, remember some of the AUX functions are only available from the Handheld. The following diagram shows how you could quickly access the AUX function from the Handheld.

CLR AUX

```
AUX FUNCTION SELECTION
AUX 1* OPERATING MODE
```

Use NXT or PREV to cycle through the menus

NXT

```
AUX FUNCTION SELECTION
AUX 2* RLL OPERATIONS
```

Press ENT to select sub-menus

ENT

```
AUX 2* RLL OPERATIONS
AUX 21 CHECK PROGRAM
```

You can also enter the exact AUX number to go straight to the sub-menu.

Enter the AUX number directly

AUX 2 1 ENT

```
AUX 2* RLL OPERATIONS
AUX 21 CHECK PROGRAM
```

AUX 1* — Operating Modes

AUX 11
Go to Run Mode

You can use AUX 11 to place the CPU in Run Mode. In Run Mode the CPU executes the program and updates the I/O modules. You can also change the PLC mode from within ***Direct***SOFT by using the PLC/PLC Modes sub-menu.

AUX 12
Go to Test Mode

Test Mode allows the CPU to start in TEST-PGM mode, enter TEST-RUN mode, run a fixed number of scans (from 1 to 65,535), and then return to TEST-PGM mode. You can also change enter TEST Mode from within ***Direct***SOFT by using the **PLC > PLC Modes** sub-menu. You gain some advantages by using Test Mode.

- The status displays are more detailed with the Handheld Programmer.
- You can enable the CPU to hold output states.

For some instructions, the Handheld Programmer TEST-RUN mode display is more detailed than the status displays shown in RUN mode. The following diagram shows an example of a Timer instruction display during TEST-RUN mode.

RUN Mode

```
$    3                  ▮
TMR    T 000    K0500
```

TEST-RUN Mode

```
$    3   T=510          ▮
TMR    T 000    K0500
```

Current Value

AUX 13
Go to Program Mode

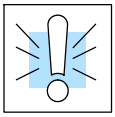
Use AUX 13 to place the CPU in Program Mode. In PGM mode, you can enter or change the program. The CPU does not execute the program or update the output modules. All output points are turned off. You can also change enter Program Mode from within ***Direct***SOFT by using the **PLC > PLC Modes** sub-menu.

AUX 14
Run Time Edit

With the DL440 or DL450 CPU you can edit programs during Run Mode and Test-Run Mode with AUX14, or from within ***Direct***SOFT by using the **PLC > PLC Modes** sub-menu.

Most of the things you can do in Program Mode also apply. For example, you can use the same techniques to search for a specific instruction or a specific address. However, you cannot use Search and Replace functions during Run Mode.

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



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

For major program changes, we strongly recommended switching the system to program mode, taking the same precautions as if it were the initial machine startup.

AUX 2* — RLL Operations

AUX 21
Check Program

Both the Handheld and ***Direct***SOFT automatically check for errors during program entry. However, there may be occasions when you want to check a program that has already been in the CPU. There are two types of checks available.

- Syntax
- Duplicate References

The Syntax check will find a wide variety of programming errors, such as missing END statements, incomplete FOR/NEXT loops, etc. If you perform this check and get an error, see Appendix B for a complete listing of programming error codes. Correct the problem and then continue running the Syntax check until the message "NO SYNTAX ERROR appears.

Use the Duplicate Reference check to verify you have not used the same output coil reference more than once. Note, this AUX function will also find the same outputs even if they have been used with the OROUT instruction, which is perfectly acceptable.

This AUX function is available on the **PLC > Diagnostics** sub-menu from within ***Direct***SOFT.

AUX 22
Change Reference

There will probably be times when you need to change an I/O address reference or control relay reference. AUX 22 allows you to quickly and easily change all occurrences of a specific instruction. For example, you can replace every instance of X5 with X10.

AUX 23
Clear Ladder Range

There have been many times when we've taken existing programs and added or removed certain portions to solve new application problems. By using AUX 23 you can select and delete a portion of the program. ***Direct***SOFT does not have a menu option for this AUX function, but you can just select the appropriate portion of the program and cut it with the editing tools.

AUX 24
Clear Ladders

AUX 24 clears the entire program from CPU memory. Before you enter a new program, you should always clear ladder memory. This AUX function is available on the **PLC > Clear PLC Memory** sub-menu within ***Direct***SOFT.

AUX 25
Select MC or Flash Memory

AUX 25 lets you select either the (optional) Memory Cartridge or the CPU's built-in flash memory as the current working memory. NOTE: operations such as Clear Ladder Range and Clear Ladders operate on the current selected memory device.

AUX 26
Copy MC Contents to Flash

AUX 26 copies the entire Memory Cartridge contents to the CPU's built-in flash memory. Any data in the flash memory will be over-written.

AUX 27
Copy Flash Contents to MC

AUX 27 copies the entire CPU's built-in flash memory contents to the Memory Cartridge memory. Any data in the Memory Cartridge will be over-written.

AUX 28
Verify Flash Contents = MC

AUX 28 verifies that the data in the CPU's internal flash memory is the same as the Memory Cartridge contents. This is useful when verifying ladder program updates, and generally managing the ladder program during program development.

AUX 3* — V-memory Operations

AUX 31
Clear V-Memory

AUX 31 clears all the information from the V-memory locations available for general use. This AUX function is available on the **PLC > Clear PLC Memory** sub-menu within ***Direct***SOFT.

AUX 32
Clear V-memory Range

Just as you can use AUX 23 to delete a portion of the control program, you can also use AUX 32 to clear a portion of V-memory. ***Direct***SOFT provides access to this AUX function on the **Tools > Memory Editor** sub-menu.

AUX 33
Find V-memory Value

AUX 33 allows you to search through a range of V-memory locations to find a specific value. For example, you may want to search a value of 1234 through the range of V1400 - V2000. This feature is also available within ***Direct***SOFT via the **Tools > Memory Editor** sub-menu.

AUX 4* — I/O Configuration

AUX 41
Show I/O Configuration

This AUX function allows you to display the current I/O configuration. With the Handheld Programmer, you can scroll through each base and I/O slot to view the complete configuration. The configuration shows the type of module installed in each slot. ***Direct***SOFT provides the same information, but it is much easier to view because you can view a complete base on one screen.

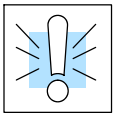
AUX 42
I/O Diagnostics

This is probably one of the most useful AUX function available in the DL405 system. This AUX function will show you the exact base and slot location of any I/O module error that has occurred. This feature is also available within ***Direct***SOFT under the **PLC > Diagnostics** sub-menu.

AUX 44
Power-up Configuration Check

By selecting this feature you can quickly detect any changes that may have occurred while the power was disconnected. For example, if someone placed an output module in a slot that previously held an input module, the configuration check would detect the change.

If the system detects a change in the I/O configuration at power-up, an error code E252 NEW I/O CONFIGURATION will be generated. You can use AUX 42 to determine the exact base and slot location where the change occurred.

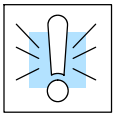


WARNING: You should always correct any I/O configuration errors before you place the CPU into RUN mode. Uncorrected errors can cause unpredictable machine operation that can result in a risk of personal injury or damage to equipment.

This feature is also available within ***Direct***SOFT under the **PLC > Setup** sub-menu.

AUX 45 Select Configuration

Even though the CPU can automatically detect configuration changes, you may actually want the new I/O configuration to be used. For example, you may have intentionally changed a module to use with a new program. You can use AUX 45 to select the new configuration, or, keep the existing configuration that is stored in memory. This feature is also available within ***Direct***SOFT from the **PLC > Setup** sub-menu.



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

AUX 46 I/O Configuration

You will probably never need to use this feature, but the DL440 CPU allows you to use AUX 46 to manually assign I/O addresses for any or all I/O slots on the local or expansion bases. It is generally much easier to do the I/O configuration operations from within ***Direct***SOFT. The software package provides a really nice screen that is available from the **PLC > Configure I/O** sub-menu.

This feature is useful if you have a standard configuration you must sometimes change slightly to accommodate special requests. For example, you may require two adjacent input modules to have addresses starting at X10 and X200 respectively.

In automatic configuration, the addresses were assigned on 8-point boundaries. Manual configuration assumes that all modules are at least 16 points, so you can only assign addresses that are a multiple of 20 (octal). For example, X30 and Y50 would not be valid starting addresses for a module. X20 and Y40 are valid examples of starting addresses in a manual configuration. This does not mean you can only use 16 or 32 point modules with manual configuration. You can use 8 point modules, but 16 addresses will be assigned and 8 of them are unused.

WARNING: If you manually configure an I/O slot, the I/O addressing for the other modules will change. This is because the DL405 products do not allow you to assign duplicate I/O addresses. You should always correct any I/O configuration errors before you place the CPU into RUN mode. Uncorrected errors can cause unpredictable machine operation that can result in a risk of personal injury or damage to equipment.

Once you have manually configured the addresses for an I/O slot, the system will automatically retain these values even after a power cycle. You can remove any manual configuration changes by simply performing an automatic configuration.

AUX 47 Intelligent I/O Monitor

This AUX function allows you to monitor the shared RAM data that is associated with many specialty I/O modules. For example, you could use this AUX function to monitor the data accessed by shared RAM in the High Speed Counter. ***Direct***SOFT provides this capability through the **PLC > Setup Global I/O** sub-menu.

AUX 5* — CPU Configuration

AUX 51
Modify Program Name

The DL405 products can use program names for memory cartridges or cassette tapes. Program names are especially useful with cassette tapes since they can store multiple programs. The program name can be up to eight characters in length and can use any of the available characters (A–Z, 0–9). AUX 51 allows you to enter a program name. You can also perform this operation from within ***Direct***SOFT by using the **PLC > Setup** sub-menu.

AUX 52
Display / Change Calendar

The DL440 CPU has a clock and calendar feature. If you are using this, you can use the Handheld and AUX 52 to set the time and date. The following format is used.

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

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

You can also perform this operation from within ***Direct***SOFT by using the **PLC > Setup** sub-menu.

AUX 53
Display Scan Time

AUX 53 displays the current, minimum, and maximum scan times. The minimum and maximum times are the ones that have occurred since the last Program Mode to Run Mode transition. You can also perform this operation from within ***Direct***SOFT by using the **PLC > Diagnostics** sub-menu.

AUX 54
Initialize Scratchpad

The DL405 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.

NOTE: You may never have to use this feature unless you have made changes that affect system memory. Usually, you’ll only need to initialize the system memory if you are changing programs and the old program required a special system setup. You can usually change from program to program without ever initializing system memory.

AUX 54 resets the system memory to the default values. You can also perform this operation from within ***Direct***SOFT by using the **PLC > Setup** sub-menu.

AUX 55
Set Watchdog Timer

The DL405 CPUs have a “watchdog” timer that is used to monitor the scan time. The default value set from the factory is 200 ms. If the scan time exceeds the watchdog time limit, the CPU automatically leaves RUN mode and enters PGM mode. The Handheld displays the following message E003 S/W TIMEOUT when the scan overrun occurs.

Use AUX 55 to increase or decrease the watchdog timer value. You can also perform this operation from within ***Direct***SOFT by using the **PLC > Setup** sub-menu.

AUX 56
CPU Network Address

Since the DL405 CPUs have built-in ***Direct***NET ports, you can use the Handheld to set the network address for the port and the port communication parameters. The default settings are:

- Station address 1
- Hex mode
- Odd parity

The ***Direct***NET Manual provides additional information about communication settings required for network operation.

NOTE: You will only need to use this procedure if you have the bottom port connected to a network. Otherwise, the default settings will work just fine.

Use AUX 56 to set the network address and communication parameters. You can also perform this operation with ***Direct***SOFT by using the **PLC > Setup** sub-menu.

AUX 57
Set Retentive Ranges

The DL405 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	DL430		DL440		DL450	
	Default Range	Avail. Range	Default Range	Avail. Range	Default Range	Avail. Range
Control Relays	C600 – C737	C0 – C737	C600 – C737	C0 – C1777	C1000 – C3777	C0 – C3777
V–Memory	V2000 – V7377	V0 – V7377	V2000 – V7777	V0 – V17777	V1400 – V37777	V0 – V37777
Timers	None by default	T0 – T177	None by default	T0 – T377	None by default	T0 – T377
Counters	CT0 – CT177	CT0 – CT177	CT0 – CT177	CT0 – CT177	CT0 – CT377	CT0 – CT377
Stages	None by default	S0 – S577	None by default	S0 – S1777	None by default	S0 – S1777

Use AUX 57 to change the retentive ranges. You can also perform this operation from within ***Direct***SOFT by using the **PLC > Setup** sub-menu.

AUX 58
Test Operations

In normal RUN mode, the outputs are turned off when you return to PGM mode. In TEST-RUN mode you can set each individual output to either turn off or hold its last output state on the transition to TEST-PGM mode. The ability to hold the output states is especially useful, since It allows you to maintain key system I/O points for examination. The following diagram shows the differences between RUN and TEST-RUN modes.

You can use AUX 58 to configure each individual output. You can also perform this operation from within ***Direct***SOFT by using the **PLC > Setup** sub-menu.

AUX 5C
Display Error History

The DL440 and DL450 CPUs will automatically log any system error codes and custom messages created with the FAULT instructions. The CPU logs the error code, date, and time the error occurred in two separate tables:

- **Error Code Table** – the system logs up to 32 errors in the table. When an error occurs, the errors already on the table are pushed down and the most recent error is loaded into the top slot. If the table is full when an error occurs, the oldest error is pushed out (erased) of the table.

- **Message Table** – the system logs up to 16 messages in this table. When a message is triggered, the messages already stored in the table are pushed down and the most recent message is loaded into the top slot. If the table is full when an error occurs, the oldest message is pushed out (erased) of the table. Example messages below.

Date	Time	Message
1993-05-26	08:41:51:11	* Conveyor-2 stopped
1993-04-30	17:01:11:56	* Conveyor-1 stopped
1993-04-30	17:01:11:12	* Limit SW1 failed
1993-04-28	03:25:14:31	* Saw Jam Detect

You can use AUX Function 5C to show the error codes or messages You can also view the errors and messages from within ***Direct***SOFT by using the **PLC > Diagnostics** sub-menu.

AUX 5D Select PLC Scan Mode

The DL450 CPU has two program scan modes: fixed and variable. In fixed mode, the scan time is lengthened to the time you specify (in milliseconds). In variable scan mode, the CPU begins each scan as soon as the previous scan's activities complete.

AUX 6* — Handheld Programmer Configuration

AUX 61 Show Revision Numbers

Over the years there have been several additions and enhancements made to the products that are compatible with the DL405 family. Sometimes these new features only work with certain releases of firmware. By using AUX 61 you can quickly view the CPU, Gate Array, and Handheld Programmer firmware revision numbers. This information is also available from within ***Direct***SOFT from the **PLC > Diagnostics** sub-menu.

AUX 62 Beeper On / Off

The Handheld has a beeper that provides confirmation of keystrokes. This can be quite annoying in an office environment. You can use Auxiliary (AUX) Function 62 to turn off the beeper.

AUX 63 Turning Off the Backlight

If necessary, you can use AUX 63 to turn off the display backlight. However, in most cases it seems to be easier to read if you just leave the display backlight enabled.

AUX 64 Select Online / Offline

You can use AUX 64 to take the Handheld Programmer offline. That is, the Handheld will not communicate with the CPU until it is returned to online operation. For example, if you have selected offline you cannot access the CPU.

AUX 65 Run Self Diagnostics

If you think the Handheld Programmer is not operating correctly, you can use AUX 65 to run a self diagnostics program. You can check the following items.

- Keypad
- Display
- LEDs and Backlight
- Tape Port (requires adapter)
- Memory Cartridge

AUX 7* — Memory Cartridge Operations

Transferrable Memory Areas

Many of these AUX functions allow you to copy different areas of memory to and from the CPU, memory cartridge, and cassette tapes. The following table shows the areas that may be mentioned.

Option and Memory Type	DL440 Range	DL430 Range
1:PGM — Program	$00000 - $07679 (7.5K program memory) $00000 - $015871 (15.5K program memory)	$00000 - $03583
2:V — V-memory	$00000 - $37777	$00000 - $07777
3:SYS — System memory	Non-selectable copies all system parameters	

AUX 71 CPU to Memory Cartridge

AUX 71 copies information from a CPU to a memory cartridge that is installed in the Handheld Programmer. If a memory cartridge is not present in the Handheld, you can just remove the memory cartridge from the CPU and place it in the Handheld, but if you want to keep the CPU that is already installed up and running you should use this AUX function to copy the CPU memory.

You can copy different portions of CPU memory to the memory cartridge as shown in the previous table.A single memory cartridge cannot hold an entire system. You may have to use more than one cartridge. If so, put V memory on a cartridge by itself.

AUX 72 Memory Cartridge to CPU

AUX 72 copies information from a memory cartridge installed in the Handheld Programmer to the CPU. If a memory cartridge is not present in the CPU, you can just remove the memory cartridge from the Handheld and place it in the CPU.You can copy different information types from the memory cartridge as shown in the previous table.

AUX 73 Compare Memory Cartridge to CPU

AUX 73 compares the Handheld memory cartridge program with the CPU program.You can compare different types of information as shown previously.

AUX 74 Memory Cartridge Blank Check

AUX 74 allows you to check the memory cartridge to make sure it is blank. It's a good idea to use this function anytime you start to copy information to a memory cartridge.

AUX 75 Clear Memory Cartridge

AUX 75 allows you to clear all data from a memory cartridge. This is true for the RAM and EEPROM memory cartridges. The UVPROM cartridges must be erased with a UV light source.

AUX 76 Display Memory Cartridge Type

You can use AUX 76 to quickly display the type of memory cartridge that is installed in both the CPU and the Handheld Programmer. Remember, there are three types of memory cartridges, CMOS RAM, EEPROM, and UVPROM.

AUX 77 Tape to Memory Cartridge

Use AUX 77 to read a program from a cassette tape. Before you begin the procedure make sure you review the procedures identified in the DL405 Handheld Programmer manual. These procedures are important to help ensure a successful transfer of information.

AUX 78
Memory Cartridge to Tape

Since it is very easy to store multiple programs on a single cassette it is very important idea to name each program. You already know you can enter a name for the CPU program. The cassette program name does not have to be the same. For example, the CPU name may be PRESS1 and the tape name may be STATION1.

Since there are three areas of CPU memory that can be transferred, it is a good idea to give each of these a separate program name. For example, you could use three programs, STAT1PGM, STAT1V, and STAT1SYS, for Station 1.

NOTE: Remember tape programs are stored sequentially. It is very easy to overwrite existing programs if you do not position the tape correctly. Make sure you review the procedures identified in the DL405 Handheld Programmer manual before beginning this procedure.

AUX 79
Compare Memory Cartridge to Tape

Use AUX 79 to compare a cassette program to a program stored in the CPU. Make sure you review the procedures identified in the DL405 Handheld Programmer manual before beginning this procedure.

AUX 8* — Password Operations

AUX 81
Modify Password

You can use AUX 81 with the DL440 or DL450 CPU to provide an extra measure of protection by entering a password that prevents unauthorized machine operations. The password must be an eight-character numeric (0–9) code. Once you've entered a password, you can remove it by entering all zeros (00000000). (This is the default from the factory.)

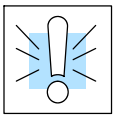
NOTE: The 440 and 450 CPUs support multi-level passwords. This allows password protection of the ladder program while not locking the communication port to operator interfaces. The multi-level password can be invoked by making a password with an upper case "A" followed by 7 numeric characters (e.g. A12345678).

The password is stored in the memory cartridge. If you install the memory cartridge in another CPU or Handheld, the password protection remains in effect.

Once you've entered a password, you can lock the CPU against access. There are two ways to lock the CPU with the Handheld Programmer.

- The CPU is always locked after a power cycle (if a password is present).
- You can use AUX 83 and AUX 84 to lock and unlock the CPU.

You can also enter or modify a password from within ***Direct***SOFT by using the **PLC > Password** sub-menu. This feature works slightly differently in ***Direct***SOFT. Once you've entered a password, the CPU is automatically locked when you exit the software package. It will also be locked if the CPU is power cycled.



WARNING: Make *sure* you remember the password *before* you lock the CPU. Once the CPU is locked you cannot view, change, or erase the password. You also cannot erase the memory cartridge and start over. You can have the password removed by returning the CPU or memory cartridge to the factory for password removal. It is the policy of AutomationDirect to clear the PLC memory along with the password, so be sure to have a copy of the program to reload into the CPU memory once the CPU or memory cartridge has been returned.

AUX 82
Unlock CPU

AUX 82 can be used to unlock a CPU that has been password protected. ***Direct***SOFT will automatically ask you to enter the password if you attempt to communicate with a CPU that contains a password.

AUX 83
Lock CPU

AUX 83 can be used to lock a CPU that contains a password. Once the CPU is locked, you will have to enter a password to gain access. Remember, this is not necessary with ***Direct***SOFT since the CPU is automatically locked whenever you exit the software package.

DL405 Error Codes

B

In This Appendix. . . .

— Error Code Table

Error Code	Name	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	This error will occur if the program scan time exceeds the time allotted to the watchdog timer. 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 using AUX 55 extend the time allotted to the watchdog timer.
E004	INVALID INSTRUCTION (DL440)	The application program has changed for some reason. SP44 will be on and the error code will be stored in V7755. This problem may possibly be due to electrical noise. Use AUX21 to check the program syntax and correct where necessary or clear the memory and re-download the program. Correct any grounding problems. If the error returns replace the CPU.
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.
E042	NO CPU BATTERY (DL450)	The CPU battery is not installed. SP43 will be on and the error code will be stored in V7757.
E043	MEMORY CARTRIDGE BATTERY LOW (DL440/DL450)	The Memory Cartridge battery is low and should be replaced. SP43 will be on and the error code will be stored in V7757.
E044	NO MEMORY CARTRIDGE BATTERY (DL450)	The Memory Cartridge 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.
E101	CPU MC MISSING (DL440/DL450)	The CPU Memory Cartridge has failed or is missing. SP44 will be on and the error code will be stored in V7755. Install or replace the Memory Cartridge.
E104	WRITE FAILED (DL440/DL450)	A write to the CPU Memory Cartridge was not successful. The Memory Cartridge may be write protected. Disassemble and check the jumper. If the error still occurs replace the Memory Cartridge.
E151	INVALID 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 be due to electrical noise (correct any grounding problems). Clear the memory and re-download the program. If the error returns replace the Memory Cartridge or CPU.
E155	RAM FAILURE	A checksum error has occurred in the system RAM. SP44 will be on and the error code will be stored in V7755. This problem may possibly be due to a low battery, electrical noise or a CPU RAM failure. Clear the memory and re-download the program. Correct any grounding problems. If the error returns replace the CPU.
E2**	I/O MODULE FAILURE	An I/O module has failed. Run AUX42 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.

Error Code	Name	Description
E202	MISSING I/O MODULE	An I/O module has failed to communicate with the CPU or is missing from the base. SP45 will be on and the error code will be stored in V7756. Run AUX42 to determine the slot and base location of the module reporting the error.
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. Run AUX42 to determine the slot and base location of the module reporting the error.
E206	USER 24V POWER SUPPLY FAILURE.	The 24VDC 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 (DL440)	A short duration power drop-out occurred on the main power line supplying power to the base.
E250	COMM. FAILURE IN THE I/O CHAIN	A failure has occurred in the local 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. Run AUX42 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 run AUX45 to accept the new configuration. SP47 will be on and the error code will be stored in V7755.
E261	I/O ADDRESS CONFLICT (DL440/DL450)	Overlapping addresses have been assigned while manually configuring the I/O. Correct the address assignments using AUX46. SP45 will be on and the error code will be stored in V7755.
E262	I/O OUT OF RANGE	An out of range I/O address has been encountered in the application program. Correct the invalid address in the program. SP45 will be on and the error code will be stored in V7755.
E263	CONFIGURED I/O ADDRESS OUT OF RANGE (DL440/DL450)	Out of range addresses have been assigned while manually configuring the I/O. Correct the address assignments using AUX46. SP45 will be on and the error code will be stored in V7755.
E264	DUPLICATE I/O REFERENCE (DL440/DL450)	Duplicate addresses have been assigned while manually configuring the I/O. Correct the address assignments using AUX46.
E311	HP COMM ERROR 1	A request from the handheld programmer could not be processed by the CPU. Clear the error and retry the request. If the error continues, replace the CPU. SP46 will be on and the error code will be stored in V7756.
E312	HP COMM ERROR 2	A data error was encountered during communications with the CPU. Clear the error and retry the request. If the error continues, check the cabling between the two devices, replace the handheld programmer; then if necessary replace the CPU. SP46 will be on and the error code will be stored in V7756.

Error Code	Name	Description
E313	HP COMM ERROR 3	An address error was encountered during communications with the CPU. Clear the error and retry the request. If the error continues, check the cabling between the two devices, replace the handheld programmer; then if necessary replace the CPU. SP46 will be on and the error code will be stored in V7756.
E316	HP COMM ERROR 6	A mode error was encountered during communications with the CPU. Clear the error and retry the request. If the error continues, replace the handheld programmer; then if necessary replace the CPU. SP46 will be on and the error code will be stored in V7756.
E320	HP COMM TIME-OUT	The CPU did not respond to the handheld programmer communication request. Check to ensure cabling is correct and not defective. Power cycle the system. If the error continues, replace the CPU first and then the handheld programmer if necessary.
E321	COMM ERROR	A data error was encountered during communication with the CPU. Check to ensure cabling is correct and not defective. Power cycle the system. If the error continues, replace the CPU first and then the handheld programmer if necessary.
E352	BACKGROUND COMM ERROR	Communications error between CPU and intelligent module. Incorrect slot reference while attempting to use the READ/WRITE commands, such as from DCM interface. The slot number of the module which has the I/O error is stored in V7660–V7764. You must power cycle the PLC to clear this error.
E360	HP PERIPHERAL PORT TIME-OUT	The device connected to the peripheral port did not respond to the handheld programmer communication request. Check to ensure cabling is correct and not defective. The peripheral device or handheld programmer could be defective.
E4**	NO PROGRAM	A syntax error exist in the application program. The most common is a missing END statement. Run AUX21 to determine which one of the E4** series of errors is being flagged. SP52 will be on and the error code will be stored in V7755.
E401	MISSING END STATEMENT	All application programs must terminate with an END statement. Enter the END statement in appropriate location in your program. SP52 will be on and the error code will be stored in V7755.
E402	MISSING LBL (DL440/DL450)	A GOTO, GTS, MOV MC or LD LBL instruction was used without the appropriate label. Refer to the programming section for details on these instructions. SP52 will be on and the error code will be stored in V7755.
E403	MISSING RET (DL440/DL450)	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 (DL440/DL450)	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 (DL440/DL450)	A FOR instruction does not have the corresponding NEXT instruction. SP52 will be on and the error code will be stored in V7755.
E406	MISSING IRT	An interrupt routine in the program does not end with the IRT instruction. SP52 will be on and the error code will be stored in V7755.
E412	SBR/LBL>64 (DL440/DL450)	There is greater than 64 SBR, LBL or DLBL instructions in the program. This error is also returned if there is greater than 128 GTS or GOTO instructions used in the program. SP52 will be on and the error code will be stored in V7755.
E413	FOR/NEXT>64 (DL440)	There is greater than 64 FOR/Next loops in the application program. SP52 will be on and the error code will be stored in V7755. (The DL450 allows unlimited FOR-NEXT usage).

Error Code	Name	Description
E421	DUPLICATE STAGE REFERENCE	Two or more SG or ISG labels exist in the application program with the same number. A unique number must be allowed for each Stage and Initial Stage. SP52 will be on and the error code will be stored in V7755.
E422	DUPLICATE SBR/LBL REFERENCE (DL440/DL450)	Two or more SBR or LBL instructions exist in the application program with the same number. A unique number must be allowed for each Subroutine and Label. SP52 will be on and the error code will be stored in V7755.
E423	NESTED LOOPS (DL440/DL450)	Nested loops (programming one FOR/NEXT loop inside of another) is not allowed in the DL440 series. SP52 will be on and the error code will be stored in V7755.
E431	INVALID ISG/SG ADDRESS	An ISG or SG must not be programmed after the end statement such as in a subroutine. SP52 will be on and the error code will be stored in V7755.
E432	INVALID JUMP (GOTO) ADDRESS (DL440/DL450)	A LBL that corresponds to a GOTO instruction must not be programmed after the end statement such as in a subroutine. SP52 will be on and the error code will be stored in V7755.
E433	INVALID SBR ADDRESS (DL440/DL450)	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 (DL440/DL450)	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 (DL440/DL450)	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 (DL440/DL450)	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 (DL440/DL450)	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	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.

Error Code	Name	Description
E462	STACK UNDERFLOW	An unmatched number of logic levels have been stored on the stack. Insure the number of AND STR and OR STR instructions match the number of STR instructions.
E463	LOGIC ERROR	A STR instruction was not used to begin a rung of ladder logic.
E464	MISSING CKT	A rung of ladder logic is not terminated properly.
E471	DUPLICATE COIL REFERENCE	Two or more OUT instructions reference the same I/O point.
E472	DUPLICATE TMR REFERENCE	Two or more TMR instructions reference the same number.
E473	DUPLICATE CNT REFERENCE	Two or more CNT instructions reference the same number.
E480	INVALID CV ADDRESS (DL440/DL450)	The CV instruction is used in a subroutine or program interrupt routine. The CV instruction may only be used in the main program area (before the END statement).
E481	CONFLICTING INSTRUCTIONS (DL440/DL450)	An instruction exists between convergence stages.
E482	MAX. CV INSTRUCTIONS EXCEEDED (DL440/DL450)	Number of CV instructions exceeds 17.
E483	INVALID CV JUMP ADDRESS (DL440/DL450)	CV JMP has been used in a subroutine or a program interrupt routine.
E484	MISSING CV INSTRUCTION (DL440/DL450)	CV JMP is not preceded by the CV instruction. A CV JMP must immediately follow the CV instruction.
E485	MISSING REQUIRED INSTRUCTION (DL440/DL450)	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 (DL440/DL450)	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 (DL440/DL450)	The CALL BLK instruction is not followed by a ST BLK instruction.
E488	INVALID ST BLK ADDRESS (DL440/DL450)	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 (DL440/DL450)	The control relay used for the ST BLK instruction is being used as an output elsewhere.

o

Error Code	Name	Description
E490	MISSING SG INSTRUCTION (DL440/DL450)	The ST BLK instruction is not immediately followed by the SG instruction.
E491	INVALID ISG INSTRUCTION ADDRESS (DL440/DL450)	There is an ISG instruction between the ST BLK and END BLK instructions.
E492	INVALID END BLK ADDRESS (DL440/DL450)	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 REQUIRED INSTRUCTION (DL440/DL450)	A [CV, SG, ISG, ST BLK, END] instruction must immediately follow the END BLK instruction.
E494	MISSING END BLK INSTRUCTION (DL440/DL450)	The ST BLK instruction is not followed by a END BLK instruction.
E499	INVALID PRINT INSTRUCTION	Invalid PRINT instruction usage. Quotation marks and/or spaces were not entered or were entered incorrectly.
E501	BAD ENTRY	An invalid keystroke or series of keystrokes was entered into the handheld programmer.
E502	BAD ADDRESS	An invalid or out of range address was entered into the handheld programmer.
E503	BAD COMMAND	An invalid instruction was entered into the handheld programmer.
E504	BAD REF/VAL	An invalid value or reference number was entered with an instruction.
E505	INVALID INSTRUCTION	An invalid instruction was entered into the handheld programmer.
E506	INVALID OPERATION	An invalid operation was attempted by the handheld programmer.
E520	BAD OP-RUN	An operation which is invalid in the RUN mode was attempted by the handheld programmer.
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	KEYSWITCH	An operation was attempted by the handheld programmer while the CPU keyswitch 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 AUX64.
E540	CPU LOCKED (DL440/DL450)	The CPU has been password locked. To unlock the CPU use AUX82 with the password.
E541	WRONG PASSWORD (DL440/DL450)	The password used to unlock the CPU with AUX82 was incorrect.

Error Code	Name	Description
E542	PASSWORD RESET (DL440/DL450)	The CPU powered up with an invalid password and reset the password to 00000000. A password may be re-entered using AUX81.
E601	MEMORY FULL	Attempted to enter an instruction which required more memory than is available in the CPU.
E602	INSTRUCTION MISSING	A search function was performed and the instruction was not found.
E603	DATA MISSING (DL440/DL450)	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 referenced an I/O module as the incorrect type of module.
E620	OUT OF MEMORY	An attempt to transfer more data between the CPU and handheld programmer than the receiving device can hold.
E621	MC NOT BLANK	An attempt to write to a non-blank Memory Cartridge was made. Erase the cartridge and then retry the write.
E622	NO HP MC	A data transfer was attempted with no Memory Cartridge or possibly a faulty Memory Cartridge in the handheld programmer.
E623	SYSTEM MC	A function was requested with a Memory Cartridge which contains system information only.
E624	V-MEMORY ONLY	A function was requested with a Memory Cartridge which contains V-memory data only.
E625	PROGRAM ONLY	A function was requested with a Memory Cartridge which contains program data only.
E626	PROM MC	An attempt to transfer data from a tape to a UVPROM Memory Cartridge. This transfer must be made using a CMOS RAM Cartridge.
E627	BAD WRITE	An attempt to write to a write-protected or faulty Memory Cartridge was made. Check the write-protect jumper inside the cartridge, then replace if necessary.
E640	COMPARE ERROR	A compare between the Memory cartridge and the source data was found to be in error. Erase the Memory Cartridge and retry the operation, replace the Memory Cartridge if necessary.
E641	VOLUME LEVEL	The volume level of the cassette player is not set properly. Adjust the volume and retry the operation.
E642	CHECKSUM ERROR	An error was detected while data was being transferred to the handheld programmer's Memory Cartridge. Check cabling and retry the operation.
E650	HP SYSTEM ERROR	A system error has occurred in the handheld programmer. Power cycle the handheld programmer. If the error returns, replace the handheld programmer.
E651	HP ROM ERROR	A ROM error has occurred in the handheld programmer. Power cycle the handheld programmer. If the error returns, replace the handheld programmer.
E652	HP RAM ERROR	A RAM error has occurred in the handheld programmer. Power cycle the handheld programmer. If the error returns, replace the handheld programmer.
E653	MC BATTERY LOW	The battery in the CMOS RAM cartridge is low and should be replaced.

C

Instruction Execution Times

In This Appendix. . . .

— Introduction
— Boolean Instructions
— Comparative Boolean Instructions
— Immediate Instructions
— Clock/Calendar Instructions
— Timer, Counter, and Shift Register Instructions
— Accumulator/Data Stack Load and Output Instructions
— Accumulator Logic Instructions
— Bit Operation Instructions
— Math Instructions
— Number Conversion Instructions
— Table Instructions
— CPU Control Instructions
— Program Control Instructions
— Interrupt Instructions
— RLL*PLUS* Instructions
— Intelligent I/O Instructions
— Network Instructions
— Message Instructions
— Drum Instructions

Introduction

This appendix contains several tables that provide the instruction execution times for the DL430, DL440, and DL450 CPUs. Many execution times depend on the type of data used with the instruction. Registers are classified into the following types:

- Data (word) Registers
- Bit Registers

V-Memory Data Registers

Some V-memory locations are considered data registers, such as timer or counter current values. Standard User V-memory is classified as a V-memory data register. Note that can load a bit pattern into these types of registers, even though their primary use is for data registers. The following locations are data registers:

Data Registers	DL430	DL440	DL450
Timer Current Values	V00000 – V 00177	V00000 – V00377	V00000 – V00377
Counter Current Values	V01000 – V01177	V01000 – V01177	V01000 – V01377
User Data Words	V1400 – V7377	V1400 – V7377 V10000 – V17777	V1400 – V7377 V10000 – V37777

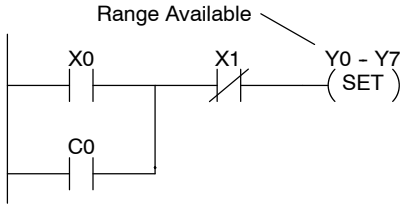
V-Memory Bit Registers

You may recall some of the discrete points such as X, Y, C, etc. are automatically mapped into V–memory. The following bit registers contain this data:

Bit Registers	DL430	DL440	DL450
Input Points (X)	V40400 – V 40423	V40400 – V 40423	V40400 – V 40477
Output Points (Y)	V40500 – V40523	V40500 – V40523	V40500 – V40577
Control Relays (C)	V40600 – V40635	V40600 – V40677	V40600 – V40777
Timer Status Bits	V41100 – V41107	V41100 – V41117	V41100 – V41117
Counter Status Bits	V41140 – V41147	V41140 – V41147	V41140 – V41157
Stages	V41000 – V41027	V41000 – V41077	V41000 – V41077
Remote I/O, (Global GX) (Global GY)	V40000 – V40037	V40000 – V40077	V40000 – V40177 V40200 – V40377

How to Read the Tables

Some instructions can have more than one parameter. For example, the SET instruction shown in the ladder program to the right can set a single point or a range of points.



In these cases, execution times depend on the amount and type of parameters. The execution time tables list execution times for various situations, as shown below:

SET	1st #:	X, Y, C, S, GX, GY	20.8 µs
	2nd #:	X, Y, C, S, GX, GY (N pt)	13.0µs+7.8µsxN
RST	1st #:	X, Y, C, S, GX, GY	19.5 µs
	2nd #:	X, Y, C, S, GX, GY (N pt)	11.7µs+7.8µsxN

Execution depends on numbers of locations and types of data used

NOTE: The GY data type listed for some instructions throughout this appendices is valid only for the DL450 CPU, which uses both GX and GY data types for Remote I/O points. The DL430 and DL440 CPUs use the GX data type for any Remote I/O points.

Boolean Instructions

Boolean Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**
STR	X, Y, C, T, CT	4.7 µs	4.7 µs	0.33 µs	0.33 µs	0.96 µs	0.96 µs
	S, SP, GX, GY	4.7 µs	4.7 µs	8.0 µs	8.0 µs	1.0 µs	1.0 µs
STRN	X, Y, C, T, CT	5.7 µs	5.7 µs	0.33 µs	0.33 µs	1.0 µs	1.0 µs
	S, SP, GX, GY	5.7 µs	5.7 µs	8.0 µs	8.0 µs	1.0 µs	1.0 µs
STRB	V:Register (Bit)	—	—	—	—	5.0 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	15.3 µs	??? µs
STRNB	V:Register (Bit)	—	—	—	—	4.9 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	15.1 µs	??? µs
OR	X, Y, C, T, CT	3.1 µs	3.1 µs	0.33 µs	0.33 µs	0.9 µs	0.9 µs
	S, SP, GX, GY	3.1 µs	3.1 µs	4.24 µs	4.24 µs	0.9 µs	0.9 µs
ORN	X, Y, C, CT	4.1 µs	4.1 µs	0.33 µs	0.33 µs	0.96 µs	0.96 µs
	S, SP, GX, GY	4.1 µs	4.1 µs	4.67 µs	4.67 µs	0.96 µs	0.96 µs
ORB	V:Register (Bit)	—	—	—	—	4.7 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	15.0 µs	??? µs
ORNB	V:Register (Bit)	—	—	—	—	4.6 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	14.8 µs	??? µs
AND	X, Y, C, T, CT	2.4 µs	2.4 µs	0.33 µs	0.33 µs	0.8 µs	0.8 µs
	S, SP, GX, GY	2.4 µs	2.4 µs	2.72 µs	2.72 µs	0.8 µs	0.8 µs
ANDN	X, Y, C, T, CT	3.4 µs	3.4 µs	0.33 µs	0.33 µs	0.9 µs	0.9 µs
	S, SP, GX, GY	3.4 µs	3.4 µs	3.14 µs	3.14 µs	0.9 µs	0.9 µs
ANDB	V:Register (Bit)	—	—	—	—	4.6 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	14.8 µs	??? µs
ANDNB	V:Register (Bit)	—	—	—	—	4.4 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	14.7 µs	??? µs
ANDSTR	None	1.8 µs	1.8 µs	0.33 µs	0.33 µs	0.5 µs	0.5 µs
ORSTR	None	1.8 µs	1.8 µs	0.33 µs	0.33 µs	0.5 µs	0.5 µs
OUT	X, Y, C	6.7 µs	6.7 µs	0.33 µs	0.33 µs	2.9 µs	2.9 µs
	GX, GY	6.7 µs	6.7 µs	2.06 µs	2.06 µs	2.9 µs	2.9 µs
OUTB	V:Register (Bit)	—	—	—	—	5.6 µs	??? µs
	P:Indirect (Bit)	—	—	—	—	16.4 µs	??? µs
OROUT	X, Y, C, GX, GY	8.3 µs	8.3 µs	6.1 µs	6.1 µs	3.4 µs	3.4 µs
NOT	None	—	—	3.2 µs	—	**???** µs	—

Boolean Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**
PD	X, Y, C	15.1 µs	15.1 µs	8.5 µs	8.5 µs	10.0 µs	10.0 µs
STRPD	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	3.7 µs	3.7 µs
STRND	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	2.8 µs	2.8 µs
ANDPD	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	3.6 µs	3.6 µs
ANDND	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	2.7 µs	2.7 µs
ORPD	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	3.6 µs	3.6 µs
ORND	X, Y, C, T, CT, S, SP, GX, GY	—	—	—	—	2.7 µs	2.7 µs
SET	1st #: X, Y, C	20.8 µs	5.2 µs	0.33 µs	0.33 µs	5.6 µs	1.0 µs
	S, GX, GY	20.8 µs	5.2 µs	14.6 µs	5.4 µs	5.6 µs	1.0 µs
	2nd #: X, Y, C, S, GX (N pt)	13.0 µs+ 7.8µsxN	5.2 µs	8.9 µs+ 5.7µsxN	5.4 µs	7.6 µs+ 0.6µsxN	1.2 µs
SETB	V:Register (Bit)	—	—	—	—	10.5 µs	2.3 µs
	P:Indirect (Bit)	—	—	—	—	22.2 µs	13.8 µs
RST	1st #: X, Y, C	19.5 µs	5.2 µs	0.33 µs	0.33 µs	5.6 µs	1.0 µs
	S, GX, GY	19.5 µs	5.2 µs	13.7 µs	4.5 µs	5.6 µs	5.6 µs
	2nd #: X, Y, C, S, GX, GY (N pt)	11.7 µs+ 7.8µsxN	5.2 µs	8.0 µs+ 5.7µsxN	4.5 µs	7.6 µs+ 0.6µsxN	1.2 µs
	1st #: T, CT	28.2 µs	5.2 µs	15.2 µs	3.8 µs	10.2 µs	0.9 µs
	2nd #: T, CT (N pt)	23.3 µs+ 5.8µsxN	5.2 µs	10.7 µs+ 4.6µsxN	3.8 µs	6.1 µs+ 1.9µsxN	1.2 µs
RSTB	V:Register (Bit)	—	—	—	—	10.5 µs	2.3 µs
	P:Indirect (Bit)	—	—	—	—	22.2 µs	13.8 µs
PAUSE	1wd: Y	24.0 µs	24.0 µs	14.7 µs	14.7 µs	7.4 µs	7.5 µs
	2wd: Y (N points)	18 µs+ 6µs x N	18µs+ 6µs x N	11µs+ 4µs x N	11µs+ 4µs x N	14.6µs+ 0.32µs xN	14.6µs+ 0.3µs x N

Comparative Boolean Instructions

Comparative Boolean (cont.)			DL430		DL440		DL450	
Instruction	**Legal Data Types**		**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
STRE	1st	2nd						
	V: Data Reg.	V:Data Reg.	61 µs	13.5 µs	56 µs	12.2 µs	4.9 µs	4.9
		V:Bit Reg.	182 µs	13.5 µs	130 µs	12.2 µs	4.9 µs	4.9
		K:Constant	75 µs	13.5 µs	69 µs	12.2 µs	3.5 µs	3.5
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9
	V: Bit Reg.	V:Data Reg.	182 µs	13.5 µs	130 µs	12.2 µs	4.9 µs	4.9
		V:Bit Reg.	300 µs	13.5 µs	206 µs	12.2 µs	4.9 µs	4.9
		K:Constant	193 µs	13.5 µs	206 µs	12.2 µs	3.5 µs	3.5
		P:Indir. (Data)	—	—	252 µs	12.2 µs	9.9 µs	9.9
		P:Indir. (Bit)	—	—	347 µs	12.2 µs	9.9 µs	9.9
	P:Indir. (Data)	V:Data Reg.	—	—	174 µs	12.2 µs	3.4 µs	9.9
		V:Bit Reg.	—	—	248 µs	12.2 µs	3.4 µs	9.9
		K:Constant	—	—	182 µs	12.2 µs	8.3 µs	8.3
		P:Indir. (Data)	—	—	296 µs	12.2 µs	14.3 µs	14.3
		P:Indir. (Bit)	—	—	387 µs	12.2 µs	14.3 µs	14.3
	P:Indir. (Bit)	V:Data Reg.	—	—	265 µs	12.2 µs	3.4 µs	9.9
		V:Bit Reg.	—	—	341 µs	12.2 µs	3.4 µs	9.9
		K:Constant	—	—	276 µs	12.2 µs	8.3 µs	8.3
		P:Indir. (Data)	—	—	387 µs	12.2 µs	14.3 µs	14.3
		P:Indir. (Bit)	—	—	480 µs	12.2 µs	14.3 µs	14.3
STRNE	1st	2nd						
	V: Data Reg.	V:Data Reg.	63 µs	13.5 µs	56 µs	12.2 µs	4.9 µs	4.9
		V:Bit Reg.	180 µs	13.5 µs	130 µs	12.2 µs	4.9 µs	4.9
		K:Constant	77 µs	13.5 µs	69 µs	12.2 µs	3.5 µs	3.5
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9
	V: Bit Reg.	V:Data Reg.	180 µs	13.5 µs	130 µs	12.2 µs	4.9 µs	4.9
		V:Bit Reg.	298 µs	13.5 µs	206 µs	12.2 µs	4.9 µs	4.9
		K:Constant	195 µs	13.5 µs	206 µs	12.2 µs	3.5 µs	3.5
		P:Indir. (Data)	—	—	252 µs	12.2 µs	9.9 µs	9.9
		P:Indir. (Bit)	—	—	347 µs	12.2 µs	9.9 µs	9.9
	P:Indir. (Data)	V:Data Reg.	—	—	174 µs	12.2 µs	3.4 µs	9.9
		V:Bit Reg.	—	—	248 µs	12.2 µs	3.4 µs	9.9
		K:Constant	—	—	182 µs	12.2 µs	8.3 µs	8.3
		P:Indir. (Data)	—	—	296 µs	12.2 µs	14.3 µs	14.3
		P:Indir. (Bit)	—	—	387 µs	12.2 µs	14.3 µs	14.3
	P:Indir. (Bit)	V:Data Reg.	—	—	265 µs	12.2 µs	3.4 µs	9.9
		V:Bit Reg.	—	—	341 µs	12.2 µs	3.4 µs	9.9
		K:Constant	—	—	276 µs	12.2 µs	8.3 µs	8.3
		P:Indir. (Data)	—	—	387 µs	12.2 µs	14.3 µs	14.3
		P:Indir. (Bit)	—	—	480 µs	12.2 µs	14.3 µs	14.3

Comparative Boolean (cont.)			DL430		DL440		DL450	
Instruction	**Legal Data Types**		**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
ORE	1st	2nd						
	V: Data Reg.	V:Data Reg.	62 µs	11.0 µs	48 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	180 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	73 µs	11.0 µs	55 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	180 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	297 µs	11.0 µs	198 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	191 µs	11.0 µs	131 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	166 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs
ORNE	1st	2nd						
	V: Data Reg.	V:Data Reg.	62 µs	11.0 µs	48 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	75 µs	11.0 µs	55 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	296 µs	11.0 µs	198 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	192 µs	11.0 µs	131 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	166 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs

Comparative Boolean (cont.)			DL430		DL440		DL450	
Instruction	**Legal Data Types**		**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
ANDE	1st	2nd						
	V: Data Reg.	V:Data Reg.	60 µs	11.0 µs	48 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	74 µs	11.0 µs	55 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	296 µs	11.0 µs	198 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	192 µs	11.0 µs	131 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	166 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs
ANDNE	1st	2nd						
	V: Data Reg.	V:Data Reg.	62 µs	11.0 µs	48 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	179 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	73 µs	11.0 µs	55 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	179 µs	11.0 µs	122 µs	10.8 µs	4.9 µs	4.9 µs
		V:Bit Reg.	297 µs	11.0 µs	198 µs	10.8 µs	4.9 µs	4.9 µs
		K:Constant	191 µs	11.0 µs	131 µs	10.8 µs	3.5 µs	3.5 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	166 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs

Comparative Boolean (cont.)			DL430		DL440		DL450	
Instruction	**Legal Data Types**		**Execute**	**Not Exe.**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**
STR	1st	2nd						
	T, CT	V:Data Reg.	65 µs	13.5 µs	56 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	182 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	75 µs	13.5 µs	63 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	66 µs	13.5 µs	56 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	184 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	79 µs	13.5 µs	63 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	184 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	800 µs	13.5 µs	206 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	193 µs	13.5 µs	139 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	252 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	347 µs	12.2 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	174 µs	12.2 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	248 µs	12.2 µs	9.9 µs	9.9 µs
		K:Constant	—	—	182 µs	12.2 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	296 µs	12.2 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	387 µs	12.2 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	265 µs	12.2 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	341 µs	12.2 µs	9.9 µs	9.9 µs
		K:Constant	—	—	276 µs	12.2 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	387 µs	12.2 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	480 µs	12.2 µs	14.3 µs	14.3 µs
STRN	1st	2nd						
	T, CT	V:Data Reg.	65 µs	13.5 µs	56 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	75 µs	13.5 µs	63 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	65 µs	13.5 µs	56 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	77 µs	13.5 µs	63 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	178 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	270 µs	12.2 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	180 µs	13.5 µs	130 µs	12.2 µs	4.8 µs	4.8 µs
		V:Bit Reg.	298 µs	13.5 µs	206 µs	12.2 µs	4.8 µs	4.8 µs
		K:Constant	195 µs	13.5 µs	139 µs	12.2 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	252 µs	12.2 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	347 µs	12.2 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	174 µs	12.2 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	248 µs	12.2 µs	9.9 µs	9.9 µs
		K:Constant	—	—	182 µs	12.2 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	296 µs	12.2 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	387 µs	12.2 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	265 µs	12.2 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	341 µs	12.2 µs	9.9 µs	9.9 µs
		K:Constant	—	—	276 µs	12.2 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	387 µs	12.2 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	480 µs	12.2 µs	14.3 µs	14.3 µs

Comparative Boolean (cont.)			DL430		DL440		DL450	
Instruction	**Legal Data Types**		**Execute**	**Not Exe.**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**
OR	1st	2nd						
	T, CT	V:Data Reg.	64 µs	11.0 µs	46 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	11.0 µs	124 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	74 µs	11.0 µs	57 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	168 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	264 µs	10.8 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	63 µs	13.5 µs	48 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	77 µs	13.5 µs	55 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	298 µs	13.5 µs	198 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	195 µs	13.5 µs	131 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	165 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs
ORN	1st	2nd						
	T, CT	V:Data Reg.	64 µs	11.0 µs	46 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	178 µs	11.0 µs	124 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	74 µs	11.0 µs	57 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	168 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	264 µs	10.8 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	63 µs	13.5 µs	48 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	77 µs	13.5 µs	55 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	298 µs	13.5 µs	198 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	195 µs	13.5 µs	131 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	165 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs

Comparative Boolean (cont.)			DL430		DL440		DL450	
Instruction	**Legal Data Types**		**Execute**	**Not Exe.**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**
AND	1st	2nd						
	T, CT	V:Data Reg.	62 µs	11.0 µs	46 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	178 µs	11.0 µs	124 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	73 µs	11.0 µs	57 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	168 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	264 µs	10.8 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	62 µs	11.0 µs	48 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	75 µs	11.0 µs	55 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	178 µs	11.0 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	296 µs	11.0 µs	198 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	192 µs	11.0 µs	131 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	165 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs
ANDN	1st	2nd						
	T, CT	V:Data Reg.	62 µs	11.0 µs	46 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	179 µs	11.0 µs	124 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	73 µs	11.0 µs	57 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	168 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	264 µs	10.8 µs	9.9 µs	9.9 µs
	1st	2nd						
	V: Data Reg.	V:Data Reg.	63 µs	13.5 µs	48 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	77 µs	13.5 µs	55 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	170 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	262 µs	10.8 µs	9.9 µs	9.9 µs
	V: Bit Reg.	V:Data Reg.	180 µs	13.5 µs	122 µs	10.8 µs	4.8 µs	4.8 µs
		V:Bit Reg.	298 µs	13.5 µs	198 µs	10.8 µs	4.8 µs	4.8 µs
		K:Constant	195 µs	13.5 µs	131 µs	10.8 µs	3.4 µs	3.4 µs
		P:Indir. (Data)	—	—	244 µs	10.8 µs	9.9 µs	9.9 µs
		P:Indir. (Bit)	—	—	340 µs	10.8 µs	9.9 µs	9.9 µs
	P:Indir. (Data)	V:Data Reg.	—	—	165 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	240 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	175 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	288 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	379 µs	10.8 µs	14.3 µs	14.3 µs
	P:Indir. (Bit)	V:Data Reg.	—	—	257 µs	10.8 µs	9.9 µs	9.9 µs
		V:Bit Reg.	—	—	333 µs	10.8 µs	9.9 µs	9.9 µs
		K:Constant	—	—	268 µs	10.8 µs	8.3 µs	8.3 µs
		P:Indir. (Data)	—	—	380 µs	10.8 µs	14.3 µs	14.3 µs
		P:Indir. (Bit)	—	—	473 µs	10.8 µs	14.3 µs	14.3 µs

Immediate Instructions

Immediate Instructions		DL430		DL440		DL450	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
STRI	X	22.0 µs	7.5 µs	15.6 µs	4.5 µs	128.0 µs	15.1 µs
STRNI	X	21.3 µs	7.5 µs	15.6 µs	4.5 µs	128.0 µs	15.3 µs
ORI	X	21.5 µs	5.2 µs	9.7 µs	4.8 µs	128.0 µs	14.8 µs
ORNI	X	20.8 µs	5.2 µs	9.7 µs	4.8 µs	128.0 µs	15.0 µs
ANDI	X	18.1 µs	5.2 µs	9.4 µs	2.1 µs	128.0 µs	14.8 µs
ANDNI	X	20.7 µs	5.2 µs	9.4 µs	2.1 µs	9.4 µs	2.1 µs
LDI	Y	—	—	—	—	406.3 µs	1.8 µs
OUTI	Y	27.3 µs	27.3 µs	18.0 µs	18.0 µs	135.0 µs	135.0 µs
OROUTI	Y	27.0 µs	27.0 µs	20.0 µs	20.0 µs	135.0 µs	23.5 µs
SETI	1st #: Y 2nd #: Y (N pt)	48.3 µs 24.6µs+ 23.7 xN	5.2 µs 5.2 µs	16.0 µs 18.0µs+ 15.9 xN	3.8 µs 3.8 µs	12.2 µs 139.7µs+ 0.6 xN	1.8 µs 2.2 µs
RSTI	1st #: Y 2nd #: Y (N pt)	47.1 µs 23.3µs+ 23.7 xN	5.2 µs 5.2 µs	15.1 µs 17.1µs+ 15.9 xN	3.8 µs 3.8 µs	12.2 µs 140.3µs+ 0.7 xN	1.8 µs 2.3 µs
LDIF	1st 2nd X K	—	—	252µs+ 16µs xN	3.8 µs	9.5 µs	2.3 µs
OUTIF	1st 2nd Y K	—	—	598µs+ 12µs xN	4.0 µs	12.5 µS	2.3 µs

Clock/Calendar Instructions

Clock/Calendar Instructions		DL430		DL440		DL450	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
DATE	V:Data Reg. V:Bit Reg.	—	—	135.0 µs 385.0 µs	4.0 µs 4.0 µs	21.3 µs 21.3 µs	1.9 µs 1.9 µs
TIME	V:Data Reg. V:Bit Reg.	—	—	121.0 µs 373.0 µs	4.0 µs 4.0 µs	13.2 µs 13.2 µs	1.9 µs 1.9 µs

Timer, Counter, and Shift Register Instructions

Timer, Counter, and Shift Register Instructions			DL430		DL440		DL450	
Instruction	**Legal Data Types**		**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
TMR	1st	2nd						
	T	V:Data Reg.	63.9 µs	29.1 µs	45.0 µs	21.0 µs	40.0 µs	23.0 µs
		V:Bit Reg.	179.5 µs	29.1 µs	119.0 µs	21.0 µs	40.0 µs	23.0 µs
		K:Constant	76.3 µs	29.1 µs	52.4 µs	21.0 µs	35.7 µs	19.5 µs
		P:Indir. (Data)	—	—	200.7 µs	21.0 µs	51.8 µs	33.9 µs
		P:Indir. (Bit)	—	—	273.4 µs	21.0 µs	51.8 µs	33.9 µs
TMRF	1st	2nd						
	T	V:Data Reg.	63.9 µs	29.1 µs	45.0 µs	21.0 µs	77.8 µs	22.6 µs
		V:Bit Reg.	179.5 µs	29.1 µs	119.0 µs	21.0 µs	77.8 µs	22.6 µs
		K:Constant	76.3 µs	29.1 µs	52.4 µs	21.0 µs	69.8 µs	19.2 µs
		P:Indir. (Data)	—	—	200.7 µs	21.0 µs	83.4 µs	37.5 µs
		P:Indir. (Bit)	—	—	273.4 µs	21.0 µs	83.4 µs	37.5 µs
TMRA	1st	2nd						
	T	V:Data Reg.	74.0 µs	50.3 µs	50.6 µs	33.8 µs	66.0 µs	26.1 µs
		V:Bit Reg.	298.7 µs	275.0 µs	199.5 µs	182.7 µs	66.0 µs	26.1 µs
		K:Constant	85.0 µs	61.2 µs	58.3 µs	42.1 µs	61.8 µs	21.7 µs
		P:Indir. (Data)	—	—	228.0 µs	205.4 µs	76.8 µs	37.3 µs
		P:Indir. (Bit)	—	—	376.7 µs	354.0 µs	76.8 µs	37.3 µs
TMRAF	1st	2nd						
	T	V:Data Reg.	74.0 µs	50.3 µs	50.6 µs	33.8 µs	74.8 µs	26.1 µs
		V:Bit Reg.	298.7 µs	275.0 µs	199.5 µs	182.7 µs	74.8 µs	26.1 µs
		K:Constant	74.0 µs	74.0 µs	58.3 µs	42.1 µs	71.0 µs	21.7 µs
		P:Indir. (Data)	—	—	228.0 µs	205.4 µs	85.7 µs	37.3 µs
		P:Indir. (Bit)	—	—	376.7 µs	354.0 µs	85.7 µs	37.3 µs
CNT	1st	2nd						
	CT	V:Data Reg.	46.2 µs	38.2 µs	33.6 µs	29.8 µs	37.9 µs	24.6 µs
		V:Bit Reg.	161.4 µs	159.4 µs	107.6 µs	103.8 µs	37.9 µs	24.6 µs
		K:Constant	58.6 µs	50.6 µs	41.0 µs	37.2 µs	35.7 µs	21.6 µs
		P:Indir. (Data)	—	—	191.4 µs	186.7 µs	40.8 µs	35.7 µs
		P:Indir. (Bit)	—	—	264.7 µs	260.0 µs	40.8 µs	35.7 µs
SGCNT	1st	2nd						
	CT	V:Data Reg.	57.3 µs	44.2 µs	41.3 µs	32.9 µs	39.3 µs	23.8 µs
		V:Bit Reg.	172.5 µs	159.4 µs	119.5 µs	105.7 µs	39.3 µs	23.8 µs
		K:Constant	69.3 µs	56.2 µs	46.5 µs	40.4 µs	33.7 µs	20.3 µs
		P:Indir. (Data)	—	—	164.7 µs	156.7 µs	47.1 µs	34.7 µs
		P:Indir. (Bit)	—	—	263.4 µs	247.4 µs	47.1 µs	34.7 µs
UDC	1st	2nd						
	CT	V:Data Reg.	90.0 µs	60.6 µs	60.0 µs	41.9 µs	45.1 µs	34.8 µs
		V:Bit Reg.	314.7 µs	285.3 µs	209.0 µs	190.8 µs	45.1 µs	34.8 µs
		K:Constant	102.2 µs	72.8 µs	58.6 µs	50.5 µs	41.4 µs	30.9 µs
		P:Indir. (Data)	—	—	210.7 µs	198.7 µs	56.4 µs	45.9 µs
		P:Indir. (Bit)	—	—	358.7 µs	340.0 µs	56.4 µs	45.9 µs
SR	C (N points to shift)		36.8µs+ 2.3µsxN	17.9 µs	25.6µs+ 1.6µsxN	17.7 µs	8.9 µs+ 0.5 µsxN	7.7 µs

Accumulator/Data Stack Load and Output Instructions

Accumulator/Data Stack Load and Output Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
LD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	102.0 µs 206.0 µs 112.0 µs 278.0 µs 391.0 µs	5.0 µs 5.0 µs 5.0 µs 5.0 µs 5.0 µs	97.0 µs 166.0 µs 110.0 µs 213.0 µs 272.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	6.4 µs 6.4 µs 12.7 µs 4.9 µs 4.9 µs	0.8 µs 0.8 µs 1.7 µs 0.8 µs 0.8 µs
LDD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	106.0 µs 488.0 µs 112.0 µs 282.0 µs 670.0 µs	5.0 µs 5.0 µs 5.0 µs 5.0 µs 5.0 µs	101.0 µs 354.0 µs 112.0 µs 216.0 µs 460.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.2 µs 7.2 µs 13.5 µs 5.1 µs 5.1 µs	0.8 µs 0.8 µs 1.7 µs 0.8 µs 0.8 µs
LDF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	87µs+ 16µs x N	4.0 µs	10.5µs+ 3.45µs xN	2.3 µs
LDA	O: (Octal constant for address)	95.0 µs	5.0 µs	90.0 µs	4.0 µs	4.9 µs	0.8 µs
LDX	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	517.0 µs 816.0 µs — —	5.0 µs 5.0 µs — —	433.0 µs 583.0 µs — —	4.0 µs 4.0 µs — —	10.0 µs 10.0 µs 19.9 µs 19.9 µs	1.7 µs 1.7 µs 1.7 µs 1.7 µs
LDSX	K: Constant	—	—	90.0µs	4.0 µs	19.0 µs	2.3 µs
LDR	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	30.3 µs 30.3 µs 26.6 µs 39.9 µs 39.9 µs	1.8 µs 1.8 µs 1.8 µs 1.8 µs 1.8 µs
OUT	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	26.0 µs 181.0 µs 286.0 µs 538.0 µs	5.0 µs 5.0 µs 5.0 µs 5.0 µs	15.4 µs 96.7 µs 189.4 µs 334.6 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	4.7 µs 4.7 µs 11.1 µs 11.1 µs	0.8 µs 0.8 µs 1.7 µs 1.7 µs
OUTD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	35.0 µs 419.0 µs 296.0 µs 777.0 µs	5.0 µs 5.0 µs 5.0 µs 5.0 µs	21.4 µs 232.5 µs 196.0 µs 471.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	5.4 µs 5.4 µs 11.7 µs 11.7 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs
OUTF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	52µs+ 12µs x N	4.0 µs	43.8 µs+ 6.2µs x N	2.3 µs
OUTX	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	551.0 µs 950.0 µs — —	5.0 µs 5.0 µs — —	356.0 µs 574.0 µs 471.0 µs 734.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	14.1 µs 14.1 µs 23.8 µs 23.8 µs	1.8 µs 1.8 µs 1.8 µs 1.8 µs
POP	None	82.0 µs	5.0 µs	88.0 µs	4.0 µs	3.7 µs	4.0 µs

Accumulator Logic Instructions

Accumulator Logic Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
AND	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	101.0 µs 206.0 µs — —	5.0 µs 5.0 µs — —	66.0 µs 136.0 µs 183.4 µs 272.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	4.9 µs 4.9 µs 11.1 µs 11.1 µs	0.8 µs 0.8 µs 1.7 µs 1.7 µs
ANDD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — — 115.0 µs	— — — — 5.0 µs	66.7 µs 323.4 µs 78.7 µs 186.0 µs 459.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.4 µs 7.4 µs 12.1 µs 3.6 µs 3.6 µs	1.7 µs 1.7 µs 1.7 µs 0.8 µs 0.8 µs
ANDF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	87µs+16µs x N	4.0 µs	3.9µs+ 3.4µs x N	2.3 µs
ANDS	None	—	—	77.4 µs	4.0 µs	5.0 µs	0.7 µs
OR	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	101.0 µs 206.0 µs — —	5.0 µs 5.0 µs — —	66.0 µs 136.0 µs 183.4 µs 272.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	4.9 µs 4.9 µs 11.1 µs 11.1 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs
ORD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — — 115.0 µs	— — — — 5.0 µs	69.4 µs 328.4 µs 78.7 µs 186.0 µs 459.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.5 µs 7.5 µs 7.5 µs 3.7 µs 3.7 µs	1.8 µs 1.8 µs 1.8 µs 0.8 µs 0.8 µs
ORF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	87µs+16µs x N	4.0 µs	8.8µs+ 3.5µs x N	2.3 µs
ORS	None	—	—	77.4 µs	4.0 µs	5.0 µs	0.7 µs
XOR	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	101.0 µs 206.0 µs — —	5.0 µs 5.0 µs — —	33.0 µs 136.0 µs 183.4 µs 272.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	5.0 µs 5.0 µs 11.2 µs 11.2 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs
XORD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — — 115.0 µs	— — — — 5.0 µs	69.4 µs 328.4 µs 78.7 µs 186.0 µs 459.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.5 µs 7.5 µs 12.1 µs 3.7 µs 3.7 µs	1.8 µs 1.8 µs 1.8 µs 0.8 µs 0.8 µs
XORF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	87µs+ 16µs x N	4.0 µs	8.8 µs+ 3.5µs x N	2.3 µs
XORS	None	—	—	77.3 µs	4.0 µs	5.0 µs	0.7 µs
CMP	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	56.0 µs 162.0 µs — —	5.0 µs 5.0 µs — —	36.0 µs 106.7 µs 158.7 µs 248.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	6.1 µs 6.1 µs 12.4 µs 12.4 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs

Accumulator Logic Instructions (cont.)		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
CMPD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — — —	— — — — —	48.7 µs 312.7 µs 67.4 µs 181.4 µs 455.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	8.4 µs 8.4 µs 4.6 µs 13.0 µs 13.0 µs	1.8 µs 1.8 µs 0.8 µs 1.8 µs 1.8 µs
CMPF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	248µs+ 16µs x N	4.0 µs	12.4µs+ 3.5µs x N	2.3 µs
CMPR	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	— — — — —	— — — — —	— — — — —	— — — — —	39.6 µs 39.6 µs 29.7 µs 49.2 µs 49.2 µs	1.8 µs 1.8 µs 1.8 µs 1.8 µs 1.8 µs
CMPS	None	100.0 µs	5.0 µs	99.0 µs	4.0 µs	5.8 µs	0.7 µs

Bit Operation Instructions

Bit Operation Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
SUM	None	280.0 µs	5.0 µs	183.3 µs	4.0 µs	12.1 µs	1.6 µs
SHFL	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	52µs+ 21µs x N 157µs+ 21µs x N 63µs+ 21µs x N	5.0 µs 5.0 µs 5.0 µs	33µs+ 14µs x N 103µs+ 14µs x N 43µs+ 14µs x N	4.0 µs 4.0 µs 4.0 µs	9.8 µs+ .16µs x N 9.8 µs+ .16µs x N 7.9 µs+ .16µs x N	1.8 µs 1.8 µs 1.8 µs
SHFR	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	57µs+ 21µs x N 163µs+ 21µs x N 69µs+ 21µs x N	5.0 µs 5.0 µs 5.0 µs	36µs+ 14µs x N 107µs+ 14µs x N 43µs+ 14µs x N	4.0 µs 4.0 µs 4.0 µs	9.8 µs+ .16µs x N 9.8 µs+ .16µs x N 7.9 µs+ .16µs x N	1.8 µs 1.8 µs 1.8 µs
ROTL	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	66µs+ 25µs x N 171µs+ 25µs x N 78µs+ 25µs x N	5.0 µs 5.0 µs 5.0 µs	42µs+ 17µs x N 112µs+ 17µs x N 51µs+ 17s x N	4.0 µs 4.0 µs 4.0 µs	5.3 µs 5.3 µs 7.1 µs	1.9 µs 1.9 µs 1.9 µs
ROTR	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	69µs+ 25µs x N 174µs+ 25µs x N 81µs+ 25µs x N	5.0 µs 5.0 µs 5.0 µs	44µs+ 17µs x N 114µs+ 17µs x N 54µs+ 17s x N	4.0 µs 4.0 µs 4.0 µs	5.2 µs 5.2 µs 7.1 µs	1.9 µs 1.9 µs 1.9 µs
ENCO	None	107.0 µs	5.0 µs	69.4 µs	4.0 µs	23.5 µs	1.6 µs
DECO	None	61.0 µs	5.0 µs	39.4 µs	4.0 µs	6.5 µs	1.6 µs

Math Instructions

Math Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
ADD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	308.0 µs 413.0 µs — —	5.0 µs 5.0 µs — —	203.4 µs 273.4 µs 318.0 µs 406.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	43.8 µs 43.8 µs 50.2 µs 50.2 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs
ADDD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	313.0 µs 694.0 µs 347.0 µs — —	5.0 µs 5.0 µs 5.0 µs — —	206.0 µs 460.7 µs 250.0 µs 374.0 µs 594.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	48.9 µs 48.9 µs 37.7 µs 53.6 µs 53.6 µs	1.8 µs 1.8 µs 0.8 µs 1.8 µs 1.8 µs
SUB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	308.0 µs 413.0 µs — —	5.0 µs 5.0 µs — —	203.4 µs 273.4 µs 317.4 µs 405.4 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	43.0 µs 43.0 µs 49.4 µs 49.4 µs	0.8 µs 0.8 µs 1.8 µs 1.8 µs
SUBD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	313.0 µs 694.0 µs 347.0 µs — —	5.0 µs 5.0 µs 5.0 µs — —	206.7 µs 460.7 µs 250.7 µs 320.0 µs 592.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	48.2 µs 48.2 µs 36.7 µs 52.8 µs 52.8 µs	1.8 µs 1.8 µs 0.8 µs 1.8 µs 1.8 µs
MUL	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	458.0 µs 558.0 µs 469.0 µs — —	5.0 µs 5.0 µs 5.0 µs — —	300.0 µs 370.7 µs 342.7 µs 1020.7 µs 1108.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	159.0 µs 159.0 µs 153.0 µs 165.0 µs 165.0 µs	1.8 µs 1.8 µs 0.8 µs 1.8 µs 1.8 µs
MULD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	— — — —	— — — —	480.1 µs 480.1 µs 484.0 µs 484.0 µs	1.9 µs 1.9 µs 1.9 µs 1.9 µs
DIV	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	6446.0 µs 6553.0 µs 6457.0 µs — —	5.0 µs 5.0 µs 5.0 µs — —	4358.0 µs 4446.7 µs 4361.4 µs 4490.7 µs 4578.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	217.0 µs 217.0 µs 211.0 µs 224.0 µs 224.0 µs	0.8 µs 0.8 µs 0.8 µs 1.8 µs 1.8 µs
DIVD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	— — — —	— — — —	4428.0 µs 4682.0 µs 4543.0 µs 4806.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	222.0 µs 222.0 µs 224.0 µs 224.0 µs	1.9 µs 1.9 µs 1.9 µs 1.9 µs
ADDB	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	143.0 µs 248.0 µs 145.0 µs — —	5.0 µs 5.0 µs 5.0 µs — —	94.0 µs 164.0 µs 100.0 µs 210.7 µs 298.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs 4.0 µs	11.6 µs 11.6.0 µs 17.8 µs 10.4 µs 10.4 µs	0.8 µs 0.8 µs 0.8 µs 1.8 µs 1.8 µs

Math Instructions (cont.)		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
ADDBD	V:Data Reg.	—	—	90.0 µs	4.0 µs	14.2 µs	0.8 µs
	V:Bit Reg.	—	—	344.7 µs	4.0 µs	14.2 µs	0.8 µs
	K:Constant	—	—	99.4 µs	4.0 µs	10.4 µs	0.8 µs
	P:Indir. (Data)	—	—	206.7 µs	4.0 µs	18.8 µs	1.8 µs
	P:Indir. (Bit)	—	—	479.4 µs	4.0 µs	18.8 µs	1.8 µs
SUBB	V:Data Reg.	143.0 µs	5.0 µs	94.0 µs	4.0 µs	11.8 µs	0.8 µs
	V:Bit Reg.	248.0 µs	5.0 µs	164.0 µs	4.0 µs	11.8 µs	0.8 µs
	K:Constant	145.0 µs	5.0 µs	100.0 µs	4.0 µs	10.3 µs	0.8 µs
	P:Indir. (Data)	—	—	210.0 µs	4.0 µs	18.1 µs	1.8 µs
	P:Indir. (Bit)	—	—	298.0 µs	4.0 µs	18.1 µs	1.8 µs
SUBBD	V:Data Reg.	—	—	90.0 µs	4.0 µs	14.1 µs	0.8 µs
	V:Bit Reg.	—	—	344.7 µs	4.0 µs	14.1 µs	0.8 µs
	K:Constant	—	—	99.4 µs	4.0 µs	10.2 µs	0.8 µs
	P:Indir. (Data)	—	—	205.4 µs	4.0 µs	18.8 µs	1.8 µs
	P:Indir. (Bit)	—	—	478.7 µs	4.0 µs	18.8 µs	1.8 µs
MULB	V:Data Reg.	123.0 µs	5.0 µs	80.7 µs	4.0 µs	5.2 µs	0.8 µs
	V:Bit Reg.	228.0 µs	5.0 µs	150.7 µs	4.0 µs	5.2 µs	0.8 µs
	K:Constant	134.0 µs	5.0 µs	92.7 µs	4.0 µs	3.8 µs	0.8 µs
	P:Indir. (Data)	—	—	198.0 µs	4.0 µs	11.4 µs	1.8 µs
	P:Indir. (Bit)	—	—	286.0 µs	4.0 µs	11.4 µs	1.8 µs
DIVB	V:Data Reg.	4889.0 µs	5.0 µs	3261.4 µs	4.0 µs	22.8 µs	0.8 µs
	V:Bit Reg.	4995.0 µs	5.0 µs	3331.4 µs	4.0 µs	22.8 µs	0.8 µs
	K:Constant	4902.0 µs	5.0 µs	3273.4 µs	4.0 µs	21.3 µs	0.8 µs
	P:Indir. (Data)	—	—	3380.0 µs	4.0 µs	29.1 µs	1.8 µs
	P:Indir. (Bit)	—	—	3468.0 µs	4.0 µs	29.1 µs	1.8 µs
ADDR	V:Data Reg.	—	—	—	—	46.8 µs	1.8 µs
	V:Bit Reg.	—	—	—	—	46.8 µs	1.8 µs
	K:Constant	—	—	—	—	33.8 µs	1.8 µs
	P:Indir. (Data)	—	—	—	—	56.5 µs	1.8 µs
	P:Indir. (Bit)	—	—	—	—	56.5 µs	1.8 µs
SUBR	V:Data Reg.	—	—	—	—	46.8 µs	1.8 µs
	V:Bit Reg.	—	—	—	—	46.8 µs	1.8 µs
	K:Constant	—	—	—	—	33.8 µs	1.8 µs
	P:Indir. (Data)	—	—	—	—	56.5 µs	1.8 µs
	P:Indir. (Bit)	—	—	—	—	56.5 µs	1.8 µs
MULR	V:Data Reg.	—	—	—	—	44.4 µs	1.8 µs
	V:Bit Reg.	—	—	—	—	44.4 µs	1.8 µs
	K:Constant	—	—	—	—	33.8 µs	1.8 µs
	P:Indir. (Data)	—	—	—	—	54.1 µs	1.8 µs
	P:Indir. (Bit)	—	—	—	—	54.1 µs	1.8 µs
DIVR	V:Data Reg.	—	—	—	—	49.0 µs	1.8 µs
	V:Bit Reg.	—	—	—	—	49.0 µs	1.8 µs
	K:Constant	—	—	—	—	38.4 µs	1.8 µs
	P:Indir. (Data)	—	—	—	—	58.3 µs	1.8 µs
	P:Indir. (Bit)	—	—	—	—	58.3 µs	1.8 µs

Math Instructions (cont.)		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
ADDBS	None	—	—	97.4 µs	4.0 µs	11.4 µs	0.7 µs
SUBBS	None	—	—	96.7 µs	4.0 µs	11.7 µs	0.7 µs
MULBS	None	—	—	92.7 µs	4.0 µs	5.1 µs	0.7 µs
DIVBS	None	—	—	3072.0 µs	4.0 µs	12.2 µs	0.7 µs
ADDF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	224µs+ 16µs x N	4.0 µs	50.1 µs+ 3.5 µs x N	2.3 µs
SUBF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	224µs+ 16µs x N	4.0 µs	49.2 µs+ 3.5 µs x N	2.3 µs
MULF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	325µs+ 8µs x N	4.0 µs	163.1 µs+ 3.4 µs x N	2.3 µs
DIVF	1st 2nd X, Y, C, S K:Constant T, CT, SP, GX, GY	—	—	4472µs+ 8µs x N	4.0 µs	20.9 µs+ 3.4 µs x N	2.3 µs
ADDS	None	321.0 µs	5.0 µs	265.0 µs	4.0 µs	46.5 µs	0.6 µs
SUBS	None	324.0 µs	5.0 µs	265.0 µs	4.0 µs	45.6 µs	0.7 µs
MULS	None	475.0 µs	5.0 µs	392.0 µs	4.0 µs	362.5 µs	0.7 µs
DIVS	None	6463.0 µs	5.0 µs	4061.0 µs	4.0 µs	501.8 µs	0.7 µs
INC	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	119.0 µs 381.0 µs — —	5.0 µs 5.0 µs — —	78.0 µs 230.0 µs 196.0 µs 372.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	22.6 µs 22.6 µs 29.0 µs 29.0 µs	0.8 µs 0.8 µs 1.9 µs 1.9 µs
DEC	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	119.0 µs 381.0 µs — —	5.0 µs 5.0 µs — —	78.0 µs 230.0 µs 210.7 µs 298.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	22.3 µs 22.3 µs 28.6 µs 28.6 µs	0.8 µs 0.8 µs 1.9 µs 1.9 µs
INCB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	41.0 µs 303.0 µs — —	5.0 µs 5.0 µs — —	26.0 µs 178.0 µs 143.4 µs 329.0 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.3 µs 7.3 µs 13.5 µs 13.5 µs	0.8 µs 0.8 µs 1.9 µs 1.9 µs
DECB	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	41.0 µs 303.0 µs — —	5.0 µs 5.0 µs — —	26.0 µs 178.0 µs 142.0 µs 330.7 µs	4.0 µs 4.0 µs 4.0 µs 4.0 µs	7.3 µs 7.3 µs 13.5 µs 13.5 µs	0.8 µs 0.8 µs 1.9 µs 1.9 µs

Math Instructions (cont.)		DL430		DL440		DL450	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
SQRT	None	—	—	—	—	197.6 µs	1.6 µs
SQRTR	None	—	—	—	—	61.96 µs	1.6 µs
SIN	None	—	—	—	—	274.7 µs	1.6 µs
SINR	None	—	—	—	—	110.96 µs	1.6 µs
COS	None	—	—	—	—	280.4 µs	1.6 µs
COSR	None	—	—	—	—	116.0 µs	1.6 µs
TAN	None	—	—	—	—	294.1 µs	1.6 µs
TANR	None	—	—	—	—	145.2 µs	1.6 µs
ASIN	None	—	—	—	—	349.6 µs	1.6 µs
ASINR	None	—	—	—	—	230.5 µs	1.6 µs
ACOS	None	—	—	—	—	355.5 µs	1.6 µs
ACOSR	None	—	—	—	—	237.8 µs	1.6 µs
ATAN	None	—	—	—	—	274.9 µs	1.6 µs
ATANR	None	—	—	—	—	155.5 µs	1.6 µs

Number Conversion Instructions

Number Conversion Instructions		DL430		DL440		DL450	
Instruction	Legal Data Types	Execute	Not Execute	Execute	Not Execute	Execute	Not Execute
BIN	None	409.0 µs	5.0 µs	107.4 µs	4.0 µs	127.3 µs	1.6 µs
BCD	None	409.0 µs	5.0 µs	152.0 µs	4.0 µs	125.3 µs	1.6 µs
INV	None	46.0 µs	5.0 µs	15.0 µs	4.0 µs	2.9 µs	1.6 µs
BCDCPL	None	—	—	232.7µs	4.0 µs	35.3µs	1.6 µs
ATH	V:Data Reg. (1 word) V:Bit Reg. (1 word)	— —	— —	1494.7µs 1910.0µs	4.0 µs 4.0 µs	14.3 µs 14.3 µs	1.9 µs 1.9 µs
HTA	V:Data Reg. (1 word) V:Bit Reg. (1 word)	— —	— —	1489.4µs 1960.7µs	4.0 µs 4.0 µs	14.3 µs 14.3 µs	1.9 µs 1.9 µs
SEG	None	101.0 µs	5.0 µs	64.0 µs	4.0 µs	6.9 µs	1.6 µs
GRAY	None	—	—	176.0 µs	4.0 µs	69.1 µs	1.6 µs
SFLDGT	None	—	—	224.0 µs	4.0 µs	21.6 µs	1.6 µs
RAD	None	—	—	—	—	175.0 µs	1.6 µs
RADR	None	—	—	—	—	42.5 µs	1.6 µs
DEG	None	—	—	—	—	176.2 µs	1.6 µs
DEGR	None	—	—	—	—	42.4 µs	1.6 µs
BTOR	None	—	—	—	—	11.1 µs	1.6 µs
RTOB	None	—	—	—	—	34.2 µs	1.6 µs

Table Instructions

Table Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**	**Execute**	**Not Execute**	**Execute**	**Not Execute**
FILL	V:Data Reg. — V:Bit Reg. K:Constant	— — — 	— — — 	596µs+ 74µs x N 904µs+ 74µs x N 608µs+ 74µs x N	4.0µs 4.0µs 4.0µs	13.6 µs+ 4.4µs x N 13.6µs+ 1.4µs x N 17.3µs+ 4.2µs x N	1.9 µs 1.9 µs 1.9 µs
FIND	V:Data Reg. — V:Bit Reg. K:Constant	— — — 	— — — 	372µs+ 85µs x N 442µs+ 85µs x N 384µs+ 85µs x N	4.0µs 4.0µs 4.0µs	23.0µs+ 3.1µs x N 23.0µs+ 3.1µs x N 20.2µs+ 3.1µs x N	1.9 µs 1.9 µs 1.9 µs
FDGT	V:Data Reg. V:Bit Reg. K:Constant	— — —	— — —	1028.6µs 1098.7µs 1066.7µs	4.0µs 4.0µs 4.0µs	25.9 µs 25.9 µs 26.2 µs	1.9 µs 1.9 µs 1.9 µs
MOV	V:Data Reg. V:Bit Reg.	— —	— —	1767.0µs 3188.0µs	4.0µs 4.0µs	24.1 µs 24.1 µs	1.9 µs 1.9 µs
TTD	V	—	—	748.7µs	4.0µs	29.8 µs	1.9 µs
RFB	V	—	—	747.4µs	4.0µs	21.0 µs	1.9 µs
STT	V K	— —	— —	722.7 µs 784.0 µs	4.0µs 4.0µs	27.7 µs 24.9 µs	1.9 µs 1.9 µs
RFT	V	—	—	1548.0µs	4.0µs	22.1 µs	1.9 µs
ATT	V K	— —	— —	2725.4µs 2734.4µs	4.0µs 4.0µs	25.0 µs 22.1 µs	1.9 µs 1.9 µs
MOVMC	Move V:Data Reg. to MC Move V:Bit Reg. to MC Move from MC to V:Data Reg. Move from MC to V:Bit Reg.	— — — —	— — — —	1332.7µs 2215.4µs 818.0µs 1394.0µs	4.0µs 4.0µs 4.0µs 4.0µs	6.0 µs 6.0 µs 22.7 µs 22.7 µs	1.9 µs 1.9 µs 1.9 µs 1.9 µs
LDLBL	K	—	—	62.0 µs	4.0 µs	7.6 µs	1.9 µs

CPU Control Instructions

CPU Control Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**
NOP	None	0	0	0	0	1 µs	1 µs
END	None	15.5 µs	15.5 µs	11.6 µs	11.6 µs	8.5 µs	8.5 µs
STOP	None	26.0 µs	5.0 µs	17.4 µs	4.0 µs	6.7 µs	1.6 µs
BREAK	None	—	—	717.0 µs	4.0 µs	15.3 µs	1.6 µs
RSTWT	None	22.0 µs	5.0 µs	14.7 µs	4.0 µs	4.0 µs	1.6 µs

Program Control Instructions

Program Control Instructions		DL430		DL440		DL450	
Instruction	Legal Data Types	Execute	Not Exe.	Execute	Not Exe.	Execute	Not Exe.
GOTO	K	—	—	18.7 µs	4.0 µs	5.1 µs	4.6 µs
LBL	K	—	—	0 µs	0 µs	0 µs	0 µs
FOR	V, K	—	—	32.7 µs	4.0 µs	59.6 µs	7.1 µs
NEXT	None	—	—	57.4 µs	4.0 µs	80.0 µs	0 µs
GTS	K	—	—	38.7 µs	4.0 µs	20.3 µs	5.2 µs
SBR	K	—	—	0 µs	0 µs	0.8 µs	0 µs
RTC	None	—	—	37.4 µs	4.0 µs	6.1 µs	6.1 µs
RT	None	—	—	35.4 µs	4.0 µs	5.1 µs	0 µs
MLS	K (1–7)	16.3 µs	16.3 µs	16.6 µs	16.6 µs	2.1 µs	2.1 µs
MLR	K (0–7)	20.0 µs	20.0 µs	13.4 µs	13.4 µs	2.0 µs	2.0 µs

Interrupt Instructions

Interrupt Instructions		DL430		DL440		DL450	
Instruction	Legal Data Types	Execute	Not Exe.	Execute	Not Exe.	Execute	Not Exe.
ENI	None	12.0 µs	5.0 µs	8.0 µs	4.0 µs	9.0 µs	1.6 µs
DISI	None	12.0 µs	5.0 µs	8.0 µs	4.0 µs	11.7 µs	1.6 µs
INT	0 (0–17)	0 µs	0 µs	0 µs	0 µs	0 µs	0 µs
IRTC	None	180.0 µs	5.0 µs	121.4 µs	4.0 µs	0.7 µs	0.7 µs
IRT	None	180.0 µs	—	120.0 µs	—	1.4 µs	—

RLL*PLUS* Instructions

RLL*PLUS* Instructions		DL430		DL440		DL450	
Instruction	Legal Data Types	Execute	Not Exe.	Execute	Not Exe.	Execute	Not Exe.
ISG	S	44.0 µs	42.0 µs	22.0 µs	20.0 µs	22.2 µs	21.0 µs
SG	S	44.0 µs	42.0 µs	22.0 µs	20.0 µs	22.2 µs	21.2 µs
JMP	S	14.0 µs	5.0 µs	10.7 µs	4.0 µs	20.7 µs	4.1 µs
NJMP	S	15.0 µs	5.0 µs	12.7 µs	4.0 µs	21.3 µs	4.5 µs
CV	S	—	—	30.0 µs	7.0 µs	13.8 µs	13.8 µs
CVJMP	S (N stages)	—	—	20 µs + 6 µs x N	7.0 µs	12.3 µs	12.3 µs
BCALL	C	—	—	12.0 µs	10.0 µs	19.4 µs	19.4 µs
BLK	C	—	—	21.0 µs	14.0 µs	18.3 µs	15.6 µs
BEND	None	—	—	6.0 µs	0 µs	7.8 µs	0 µs

Intelligent I/O Instructions

Intelligent Module Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**
RD	V:Data Reg. V:Bit Reg.	1130+ 123xN µs 1150+ 172xN µs	5.0 µs 5.0 µs	754+ 82xN µs 766+ 105xN µs	4.0 µs 4.0 µs	109.6 µs 109.6 µs	1.9 µs 1.9 µs
WT	V:Data Reg. V:Bit Reg.	1840+ 150xN µs 1875+ 180xN µs	5.0 µs 5.0 µs	896+ 110xN µs 917+ 120xN µs	4.0 µs 4.0 µs	109.8 µs 109.8 µs	1.9 µs 1.9 µs

Network Instructions

Network Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**
RX	X, Y, C, T, CT, GX,, GY SP, S V:Data Reg. V:Bit Reg.	760.0 µs 760.0 µs 780.0 µs	5.0 µs 5.0 µs 5.0 µs	488.0 µs 488.0 µs 488.0 µs	4.0 µs 4.0 µs 4.0 µs	111.8 µs 111.8 µs 111.8 µs	2.3 µs 2.3 µs 2.3 µs
WX	X, Y, C, T, CT, GX,, GY SP, S V:Data Reg. V:Bit Reg.	Source 755+ 12xN µs	5.0 µs 5.0 µs 5.0 µs	Source 503+ 8xN µs	4.0 µs 4.0 µs 4.0 µs	111.8 µs 111.8 µs 111.8 µs	2.3 µs 2.3 µs 2.3 µs
PRINT	ASCII	—	—	—	—	104.0 µs	2.2 µs

Message Instructions

Message Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**
FAULT	V:Data Reg. V:Bit Reg. K:Constant	164.0 µs 266.0 µs 158.0 µs	5.0 µs 5.0 µs 5.0 µs	107.0 µs 178.0 µs 158.7 µs	4.0 µs 4.0 µs 4.0 µs	107.3 µs 107.3 µs 98.1 µs	2.3 µs 2.3 µs 2.3 µs
DLBL	K	—	—	0 µs	0 µs	0 µs	0 µs
NCON	K	—	—	0 µs	0 µs	0 µs	0 µs
ACON	K	—	—	0 µs	0 µs	0 µs	0 µs

Drum Instructions

Drum Instructions		DL430		DL440		DL450	
Instruction	**Legal Data Types**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**	**Execute**	**Not Exe.**
DRUM	CT	—	—	—	—	340.0 µs	62.6 µs
EDRUM	CT	—	—	—	—	243.0 µs	100.0 µs
MDRMD	CT	—	—	—	—	206.0 µs	142.00 µs
MDRMW	CT	—	—	—	—	150.0 µs	94.00 µs

Special Relays

D

In This Appendix. . . .

- Startup and Real-time Special Relays
- CPU Status Relays
- System Monitoring Relays
- Accumulator Status Relays
- Communication Monitoring Relays

Startup and Real-time Special Relays

SP0	First scan	on for the first scan after a power cycle or program to run transition, and is reset to off on the second scan. It is useful when a function needs to be performed only at startup.
SP1	Always ON	provides a contact to insure a instruction is executed every scan.
SP2	Always OFF	provides a contact that is always off.
SP3	1 minute	on for 30 seconds and off for 30 seconds.
SP4	1 second	on for 0.5 second and off for 0.5 second.
SP5	100 millisecond	on for 50 ms. and off for 50 ms.
SP6	50 millisecond	on for 25 ms. and off for 25 ms.
SP7	Alternate scan	on every other scan.

CPU Status Relays

SP11	Forced run mode	on anytime the CPU keyswitch is in the RUN position.
SP12	Terminal run mode	on when the CPU keyswitch 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.
SP14	Break relay 1	on when the BREAK instruction is executed. It is off when the CPU is in any other mode.
SP15	Test program mode	on when the CPU is in the test program mode.
SP16	Terminal PGM mode	on when the CPU keyswitch is in the TERM position and the CPU is in program mode.
SP17	Forced stop mode	on anytime the CPU keyswitch 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 hardware interrupts are enabled using the ENI instruction.
SP25	CPU battery disabled	on when the CPU battery is disabled by dipswitch 1 on the rear of the CPU.
SP26	I/O update disable	(DL440/DL450) turned on by the application program, thus freezing the state of the I/O.
SP27	Selective I/O update disable	(DL440/DL450) 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 .
SP30	Dipswitch 1 status	(DL430/DL440) follows the On/Off status of dipswitch 1 on the rear of the CPU.
SP31	Dipswitch 2 status	(DL430/DL440) follows the On/Off status of dipswitch 2 on the rear of the CPU.
SP32	Dipswitch 3 status	(DL430/DL440) follows the On/Off status of dipswitch 3 on the rear of the CPU.
SP33	Dipswitch 4 status	(DL430/DL440) follows the On/Off status of dipswitch 4 on the rear of the CPU.
SP37	Scan control error	(DL450) 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 Relays

SP40	Critical error	on when a critical error such as I/O communication loss has occurred.
SP41	Warning	on when a non critical error such as a low battery has occurred.
SP43	Battery low (440/450)	on when either the memory cartridge battery or the CPU battery is low. Once detected, check V7757 for the exact error code.
SP44	Program memory error	on when a memory error such as a memory parity error or a failed memory cartridge has occurred.
SP45	I/O error	on when a I/O error such as blown fuse or 24V bad has occurred.
SP46	Communications error	on when a communications error has occurred on any of the CPU ports.
SP47	I/O configuration error	on if a 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	Math timeout relay	on if the CPU time out on a math function.
SP52	Grammatical error	on if a grammatical error has occurred either while the CPU is running or if the syntax check is run. Once detected check V7755 for a exact error code.
SP53	Math/Table pointer	on if there is 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 Relays

SP53	Math/Table pointer error	on if there is math execution error or a table pointer error.
SP60	Value less than	on when the value in the accumulator is less than the instruction value.
SP61	Value equal to	on when the value in the accumulator is equal to the instruction value.
SP62	Value greater than	on when the value in the accumulator is greater than the instruction value.
SP63	Zero	on when the result of the instruction causes the value in the accumulator to be zero.
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	when the 32 bit addition instruction results in a carry.
SP70	Sign (negative)	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	(DL450) on when the numerical value in the accumulator is a floating point number.
SP73	Overflow	on when a signed addition or subtraction results in an incorrect sign bit.
SP74	Under flow	(DL450) on when a floating point math operation results in an underflow error.
SP75	Data error	on when a BCD instruction is executed and a NON–BCD number was encountered.
SP76	Load zero	on when the value loaded into the accumulator by any instruction is zero.

Communication Monitoring Relays

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 module is busy.
- **Communication error** – on when a communications error occurs. This error automatically clears when another RX or WX instruction executes.

All DL405 CPUs		DL450 CPU only					
		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

DL405 Product Weights

E

In This Appendix. . . .

— Product Weight Table

Product Weight Table

CPUs	Weight
D4-430	28.2 oz. (800g)
D4-440	31 oz. (878g)
D4-450	29.8 oz. (844g)
D4-440DC-1	31.4 oz. (890g)
D4-440DC-2	31.4 oz. (890g)
D4-450DC-1	28.2 oz. (800g)
D4-450DC-2	28.2 oz. (800g)
Memory Cartridges	
D4-RAM-1	1.5 oz. (42g)
D4-RAM-2	1.3 oz. (38g)
D4-RNB	1.3 oz (38g)
D4-UV-1	2.1 oz. (60g)
D4-UV-2	2.1 oz. (60g)
D4-EE-2	1.4 oz. (40g)
Expansion Units	
D4-EX	22.7 oz (644g)
D4-EXDC	23.3 oz. (660g)
D4-EXDC-2	23.3 oz. (660g)
I/O Bases	
D4-04B-1, -04BNX	23.3 oz. (660g)
D4-06B-1 -06BNX	29.3 oz. (830g)
D4-08B-1, -08BNX	34.9 oz (990g)
DC Input Modules	
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)
D4-32ND3-2	6.7 oz. (190g)
D4-64ND2	7.8 oz. (220g)
AC Input Modules	
D4-08NA	8.5 oz. (240g)
D4-16NA	9.5 oz. (270g)

AC/DC Input Modules	
D4-16NE3	8.8 oz. (250g)
F4-08NE3S	9 oz. (256g)
DC Output Modules	**Weight**
D4-08TD1	8.5 oz. (240g)
F4-08TD1S	9.9 oz. (282g)
D4-16TD1	9.5 oz. (270g)
D4-16TD2	9.5 oz. (270g)
D4-32TD1	6.7 oz. (190g)
D4-32TD2	6.7 oz. (190g)
D4-64TD1	7.4 oz. (210g)
AC Output Modules	
D4-08TA	11.6 oz. (330g)
D4-16TA	12.3 oz. (350g)
Relay Output Modules	
D4-08TR	9.2 oz. (260g)
F4-08TRS-1	13.2 oz. (374g)
F4-08TRS-2	13.8 oz. (390g)
D4-16TR	10.9 oz. (310g)
Analog Modules	
D4-04AD	9.5 oz. (270g)
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)

Remote I/O	
D4-RM	8.0 oz. (228g)
D4-RS	27.1 oz. (767g)
D4-RSDC	26.8 oz. (760g)
D4-DCM	8.2 oz. (233g)
F4-MAS-MB	8.9 oz (252g)
F4-SLV-MB	8.9 oz (252g)
F4-SLV-TW	9.3oz (264g)
F4-SDN	9.1 oz (258g)
CoProcessors™	
F4-CP128-1	8.9 oz (252g)
F4-CP128-2	8.9 oz (252g)
F4-CP512	9.1 oz (258g)
F4-CP128-T	9.9 oz (281g)
F4-CP128-R	9.8 oz (278g)
Specialty Modules	
D4-INT	8.8 oz. (250g)
D4-HSC	12.3 oz. (350g)
F4-16PID	7.3 oz (207g)
F4-8MPI	12.3 oz (350g)
D4-16SIM	8.8 oz. (250g)
F4-SDS	7.4 oz. (211g)
F4-4LTC	12.7 oz. (361g)
D4-FILL	4.0 oz. (112g)
Programming	
D4-HPP	12.6 oz. (357g)

PLC Memory

F

In This Appendix. . . .

— DL405 PLC Memory

DL405 PLC Memory

When designing a PLC application, it is important for the PLC user to understand the different types of memory in the PLC. Two types of memory are used by the DL205 CPU, RAM and EEPROM. This memory can be configured by the PLC user as either retentive or non-retentive.

Retentive memory is memory that is configured by the user to maintain values through a power cycle or a PROGRAM to RUN transition. Non-retentive memory is memory that is configured by the PLC user to clear data after a power cycle or a PROGRAM to RUN transition. The retentive ranges can be configured with the handheld programmer using AUX 57 or ***Direct***SOFT (PLC Setup).

The contents of RAM memory can be written to and read from for an infinite number of times, but RAM requires a power source to maintain the contents of memory. The contents of RAM are maintained by the internal power supply (5VDC) only while the PLC is powered by an external source, normally 120VAC. When power to the PLC is turned off, the contents of RAM can be maintained by an optional battery (See page 3-12). The contents of RAM will be lost when external power is lost without battery backup.

The contents of EEPROM memory can be read from for an infinite number of times but there is a limit to the number of times it can be written to (typical specification is 100,000 writes). EEPROM does not require a power source to maintain the memory contents. It will retain the contents of memory indefinitely.

PLC user V-memory is stored in both volatile RAM and non-volatile EEPROM memory. See the memory map pertaining to the Data Word range for your particular CPU (page 3-40 to 3-42).

Data values that must be retained for long periods of time, when the PLC is powered off, should be stored in EEPROM based V-memory. Since EEPROM is limited to the number of times it can be written to, it is suggested that transitional logic, such as a one-shot, be used to write the data one time instead of on each CPU scan.

Data values that are continually changing or which can be initialized with program logic should be stored in RAM based V-memory.

G

ASCII Table

In This Appendix. . . .

— ASCII Conversion Table

DECIMAL TO HEX TO ASCII CONVERTER											
DEC	**HEX**	**ASCII**	**DEC**	**HEX**	**ASCII**	**DEC**	**HEX**	**ASCII**	**DEC**	**HEX**	**ASCII**
0	00	NUL	32	20	space	64	40	@	96	60	‘
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	“	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	’	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	TAB	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Numbering Systems

H

In This Appendix. . . .

— Introduction
— Binary Numbering System
— Hexadecimal Numbering System
— Octal Numbering System
— Binary Coded Decimal (BCD) Numbering System
— Real (Floating Point) Numbering System
— BCD/Binary/Decimal/Hex/Octal - What is the Difference?
— Data Type Mismatch
— Signed vs. Unsigned Integers
— AutomationDirect.com Products and Data Types

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 should be more aware of the binary system specifically the PLC programmer. 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 or Boolean. Like in a computer there are only two valid digits in Base 2 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 as in. Thirty-two bits or two words is a double word as in Table 1.

Double Word																															
Word																Word															
Byte								Byte								Byte								Byte							
Nibble				Nibble				Nibble				Nibble				Nibble				Nibble				Nibble				Nibble			
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	0	0

Table 1

Binary is not “natural“ for us to use since we have grown 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	32678	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Max Value	65535_{10}															

Table 2

Hexadecimal Numbering System

The binary numbering system can be difficult and cumbersome to interpret for some users. Therefore, the hexadecimal numbering system was developed as a convenience for humans since the PLC (computer) only understands pure binary. The hexadecimal system is useful because it can represent every byte (8 bits) as two consecutive hexadecimal digits. It is easier for us to read hexadecimal numbers than binary numbers.

The hexadecimal numbering system uses 16 characters (base 16) to represent values. The first ten characters are the same as our decimal system, 0–9, and the first six letters of the alphabet, A–F. Table 3 lists the first eighteen decimal numbers; 0–17 in the left column and the equivalent hexadecimal numbers are shown in the right column.

Decimal	Hex	Decimal	Hex
0	0	9	9
1	1	10	A
2	2	11	B
3	3	12	C
4	4	13	D
5	5	14	E
6	6	15	F
7	7	16	10
8	8	17	11

Table 3

Note that “10” and “11“ in hex are not the same as “10“ and “11“ in decimal. Only the first ten numbers 0–9 are the same in the two representations. For example, consider the hex number “D8AF“. To evaluate this hex number use the same method used to write decimal numbers. Each digit in a decimal number represents a multiple of a power of ten (base 10). Powers of ten increase from right to left. For example, the decimal number 365 means $3x10^2 + 6x10 + 5$. In hex each digit represents a multiple of a power of sixteen (base 16). Therefore, the hex number D8AF translated to decimal means $13x16^3 + 8x16^2 + 10x16 + 15 = 55471$. However, going through the arithmetic for hex numbers in order to evaluate them is not really necessary. The easier way is to use the calculator that comes as an accessory in Windows. It can convert between decimal and hex when in “Scientific“ view.

Note that a hex number such as “365“ is not the same as the decimal number “365“. Its actual value in decimal terms is $3x16^2$ 6x16 + 5 = 869. To avoid confusion, hex numbers are often labeled or tagged so that their meaning is clear. One method of tagging hex numbers is to append a lower case “h“ at the end. Another method of labeling is to precede the number with 0x. Thus, the hex number “D8AF“ can also be written “D8AFh“, where the lower case “h” at the end is just a label to make sure we know that it is a hex number. Also, D8AF can be written with a labeling prefix as “0xD8AF”.

Octal Numbering System

Many of the early computers used the octal numbering system for compiled printouts. Today, the PLC is about the only device that uses the Octal numbering system. The octal numbering system uses 8 values to represent numbers. The values are 0–7 being Base 8. Table 4 shows the first 32 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

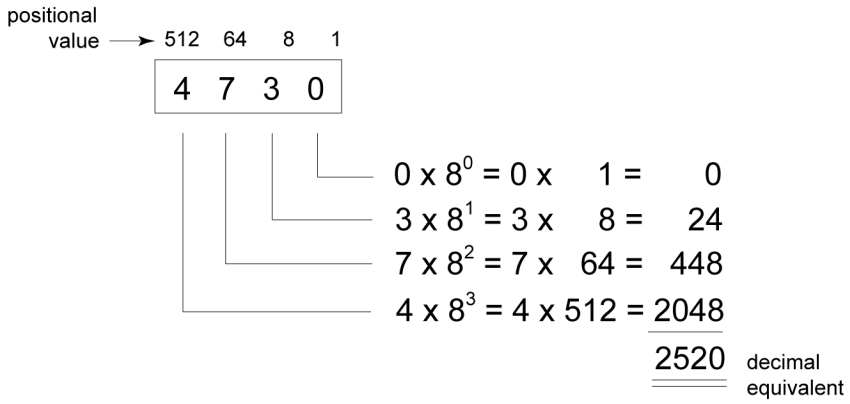
This follows the ***Direct***LOGIC PLCs. Refer to the bit maps in Chapter 3 and notice that the memory addresses are numbered in octal, as well as each bit. The octal system is much like counting in the decimal system without the digits 8 and 9 being available.

The general format for four digits of the octal number is:

$$(d \times 8^0) + (d \times 8^1) + (d \times 8^2) + (d \times 8^3)$$

where “d“ means digit. This is the same format used in the binary, decimal, or hexadecimal systems except that the base number for octal is 8.

Using the powers of expansion, the example below shows octal 4730 converted to decimal.



Binary Coded Decimal (BCD) Numbering System

BCD is a numbering system where four bits are used to represent each decimal digit. The binary codes corresponding to the hexadecimal digits A–F are not used in the BCD system. For this reason numbers cannot be coded as efficiently using the BCD system. For example, a byte can represent a maximum of 256 different numbers (i.e. 0–255) using normal binary, whereas only 100 distinct numbers (i.e. 0–99) could be coded using BCD. Also, note that BCD is a subset of hexadecimal and neither one does negative numbers.

BCD Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Power	10^3				10^2				10^1				10^0			
Bit Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Max Value	9				9				9				9			

Table 5

One plus for BCD is that it reads like a decimal number, whereas 867 in BCD would mean 867 decimal. No conversion is needed.

Real (Floating Point) Numbering System

The terms Real and floating-point both describe IEEE-754 floating point arithmetic. This standard specifies how single precision (32 bit) and double precision (64 bit) floating point numbers are to be represented as well as how arithmetic should be carried out on them. Most PLCs use the 32-bit format for floating point (or Real) numbers which will be discussed here.

Real (Floating Point 32) Bit Pattern																
Bit #	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Sign	Exponent								Mantissa						
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Mantissa (continues from above)															

Table 6

Floating point numbers, which ***Direct***LOGIC PLCs use, have three basic components: sign, exponent and mantissa. The 32 bit word required for the IEEE standard floating point numbers is shown in Table 6. It is represented as a number bits from 0 to 31, left to right. The first bit (31) is the sign bit, the next eight bits (30-23) are the exponent bits and the final 23 bits (22-0) are the fraction bits. In summary:

The sign bit is either “0” for positive or “1“ for negative;

The exponent uses base 2;

The first bit of the mantissa is typically assumed to be “1.fff“, where “f“ is the field of fraction bits.

The Internet can provide a more in-depth explanation of the floating point numbering system. One website to look at is:

http://www.psc.edu/general/software/packages/ieee/ieee.php

BCD/Binary/Decimal/Hex/Octal - What is the Difference?

Sometimes there is confusion about the differences between the data types used in ***Direct***LOGIC PLCs. The PLC's native data format is BCD, while the I/O numbering system is octal. Other numbering formats used are binary, decimal 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 in the base 2 numbering system and bit-coded data, the format of the data is dissimilar. The formats discussed are shown in Table 7 below.

Binary/Decimal Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Bit Value	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Max Value	65535															
Hexidecimal 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		
Octal 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 by 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 examples and note the same value represented by the different numbering systems. The decimal values of 67 and 4660 are used.

6 7	Decimal
0110 0111	BCD
0100 0011	Binary
4 3	Hex
1 0 3	Octal

4 6 6 0	Decimal
0100 0110 0110 0000	BCD
0001 0010 0011 0100	Binary
1 2 3 4	Hex
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

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. The following tables show a representation of 100 and a representation of -100 in this format.

Magnitude Plus Sign	
Decimal	**Binary**
100	0000 0000 0110 0100
-100	1000 0000 0110 0100

Table 11

Two's complement is a bit more complicated. 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, then a 1 is added.

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, however most of the websites deal with 8-bit examples. Two fairly good websites are listed below, you can also do a search and checkout more websites.

http://www.evergreen.edu/biophysics/technotes/program/2s_comp.htm

http://www.website.masmforum.com/tutorials/fptute/fpuchap2.htm

AutomationDirect.com Products and Data Types

DirectLOGIC PLCs

The ***Direct***LOGIC PLC family uses the octal numbering system for all addressing which includes: inputs, outputs, internal V-memory locations, timers, counters, internal control relays (bits), etc. Most data in the PLC, including timer and counter current values, is in BCD format by default. User data in V-memory locations may be stored in other data types if it is changed by the programmer, or comes from some external source, such as an operator interface. Any manipulation of data must use instructions appropriate for that data type which includes: Load instructions, Math instructions, Out box instructions, comparison instructions, etc. In many cases, the data can be changed from one data type to another, but be aware of the limitations of the various data types when doing so. For example, to change a value from BCD to decimal (binary), use a BIN instruction box. To change from BCD to a real number, use a BIN and a BTOR instruction box. When using Math instructions, ***the data types must match***. For example, a BCD or decimal (binary) number cannot be added to a real number, and a BCD number cannot be added to a decimal (binary) number. If the data types are mismatched, the results of any math operation will be meaningless.

To simplify making, number conversions Intelligent Box (IBox) Instructions are available with ***Direct***SOFT 5. These instruction descriptions are located in the DL405 IBox Instructions PLC User Manual Supplement, page 30, in the Math IBox group.

Most ***Direct***LOGIC ***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. ***Direct***LOGIC ***PID*** is another area where not all values are in BCD. In fact, nearly all of the PID parameters are stored in the PLC memory as decimal (binary) numbers.

NOTE: The PID algorithm uses magnitude plus sign for negative decimal (binary) numbers, whereas the standard math functions use two's complement. This can cause confusion while debugging a PID loop.

When using the Data View in ***Direct***SOFT, be certain that the proper format is selected for the element to be viewed. The data type and length 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".

European Union Directives (CE)

In This Appendix. . . .

— European Union (EU) Directives
— Basic EMC Installation Guidelines

European Union (EU) Directives

NOTE: The information contained in this section is intended as a guideline and is based on our interpretation of the various standards and requirements. Since the actual standards are issued by other parties and in some cases Governmental agencies, the requirements can change over time without advance warning or notice. Changes or additions to the standards can possibly invalidate any part of the information provided in this section.

This area of certification and approval is absolutely vital to anyone who wants to do business in Europe. One of the key tasks that faced the EU member countries and the European Economic Area (EEA) was the requirement to harmonize several similar yet distinct standards together into one common standard for all members. The primary purpose of a harmonized standard was to make it easier to sell and transport goods between the various countries and to maintain a safe working and living environment. The Directives that resulted from this merging of standards are now legal requirements for doing business in Europe. Products that meet these Directives are required to have a CE mark to signify compliance.

Member Countries

As of January 1, 2007, the members of the EU are Austria, Belgium, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, and the United Kingdom. Iceland, Liechtenstein, and Norway together with the EU members make up the European Economic Area (EEA) and all are covered by the Directives.

Applicable Directives

There are several Directives that apply to our products. Directives may be amended, or added, as required.

Electromagnetic Compatibility Directive (EMC) — this Directive attempts to ensure that devices, equipment, and systems have the ability to function satisfactorily in its electromagnetic environment without introducing intolerable electromagnetic disturbance to anything in that environment.

Machinery Safety Directive — this Directive covers the safety aspects of the equipment, installation, etc. There are several areas involved, including testing standards covering both electrical noise immunity and noise generation.

Low Voltage Directive — this Directive is also safety related and covers electrical equipment that has voltage ranges of 50–1000VAC and/or 75–1500VDC.

Battery Directive — this Directive covers the production, recycling, and disposal of batteries.

Compliance

Certain standards within each Directive already require mandatory compliance. The EMC Directive, which has gained the most attention, became mandatory as of January 1, 1996. The Low Voltage Directive became mandatory as of January 1, 1997.

Ultimately, we are all responsible for our various pieces of the puzzle. As manufacturers, we must test our products and document any test results and/or

installation procedures that are necessary to comply with the Directives. As a machine builder, you are responsible for installing the products in a manner which will ensure compliance is maintained. You are also responsible for testing any combinations of products that may (or may not) comply with the Directives when used together. The end user of the products must comply with any Directives that may cover maintenance, disposal, etc. of equipment or various components. *Although we strive to provide the best assistance available, it is impossible for us to test all possible configurations of our products with respect to any specific Directive. Because of this, it is ultimately your responsibility to ensure that your machinery (as a whole) complies with these Directives and to keep up with applicable Directives and/or practices that are required for compliance.*

As of January 1, 1999, the DL05, DL06 DL205, DL305, and DL405 PLC systems manufactured by either Koyo Electronics Industries, FACTS Engineering or Host Engineering, when properly installed and used, conform to the Electromagnetic Compatibility (EMC) and Low Voltage Directive requirements of the following standards.

EMC Directive Standards Relevant to PLCs

EN50081-1 Generic immunity standard for residential, commercial, and light industry

EN50081-2 Generic emission standard for industrial environment.

EN50082-1 Generic immunity standard for residential, commercial, and light industry

EN50082-2 Generic immunity standard for industrial environment.

Low Voltage Directive Standards Applicable to PLCs

EN61010-1 Safety requirements for electrical equipment for measurement, control, and laboratory use.

Product Specific Standard for PLCs

EN61131-2 Programmable controllers, equipment requirements and tests. This standard replaces the above generic standards for immunity and safety. However, the generic emissions standards must still be used in conjunction with the following standards:

——EN 61000-3-2—Harmonics

——EN 61000-3-2—Fluctuations

Warning on Electrostatic Discharge (ESD)

We recommend that all personnel take necessary precautions to avoid the risk of transferring static charges to the inside of the control cabinet, and clear warnings and instructions should be provided on the cabinet exterior. Such precautions may include the use of earth straps, similar devices or the powering down of the equipment inside the enclosure before the door is opened.

Warning on Radio Interference (RFI)

This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

General Safety

External switches, circuit breakers or external fusing, are required for these devices.

The switch or circuit breaker should be mounted near the PLC equipment.

AutomationDirect is currently in the process of changing their testing procedures from the generic standards to the product specific standards.

Special Installation Manual

The installation requirements to comply with the requirements of the Machinery Directive, EMC Directive and Low Voltage Directive are slightly more complex than the normal installation requirements found in the United States. To help with this, we have published a special manual which you can order:

See Errata Sheet at the beginning of this file. Manual DA-EU-M is no longer available.

~~**DA-EU-M** – This is an EU Installation Manual that covers special installation requirements to meet the EU Directive requirements. Order this manual to obtain the most up-to-date information.~~

Other Sources of Information

Although the EMC Directive gets the most attention, other basic Directives, such as the Machinery Directive and the Low Voltage Directive, also place restrictions on the control panel builder. Because of these additional requirements it is recommended that the following publications be purchased and used as guidelines:

BSI publication TH 42073: February 1996 – covers the safety and electrical aspects of the Machinery Directive

EN 60204-1:1992 – General electrical requirements for machinery, including Low Voltage and EMC considerations

IEC 1000-5-2: EMC earthing and cabling requirements

IEC 1000-5-1: EMC general considerations

It may be possible for you to obtain this information locally; however, the official source of applicable Directives and related standards is:

~~**The Office for Official Publications of the European Communities** L-2985 Luxembourg; quickest contact is via the World Wide Web at~~
~~http://euro-op.eu.int/indexn.htm~~

~~Another source is:~~

~~British Standards Institution – Sales Department~~
~~Linford Wood~~
~~Milton Keynes~~
~~MK14 6LE~~

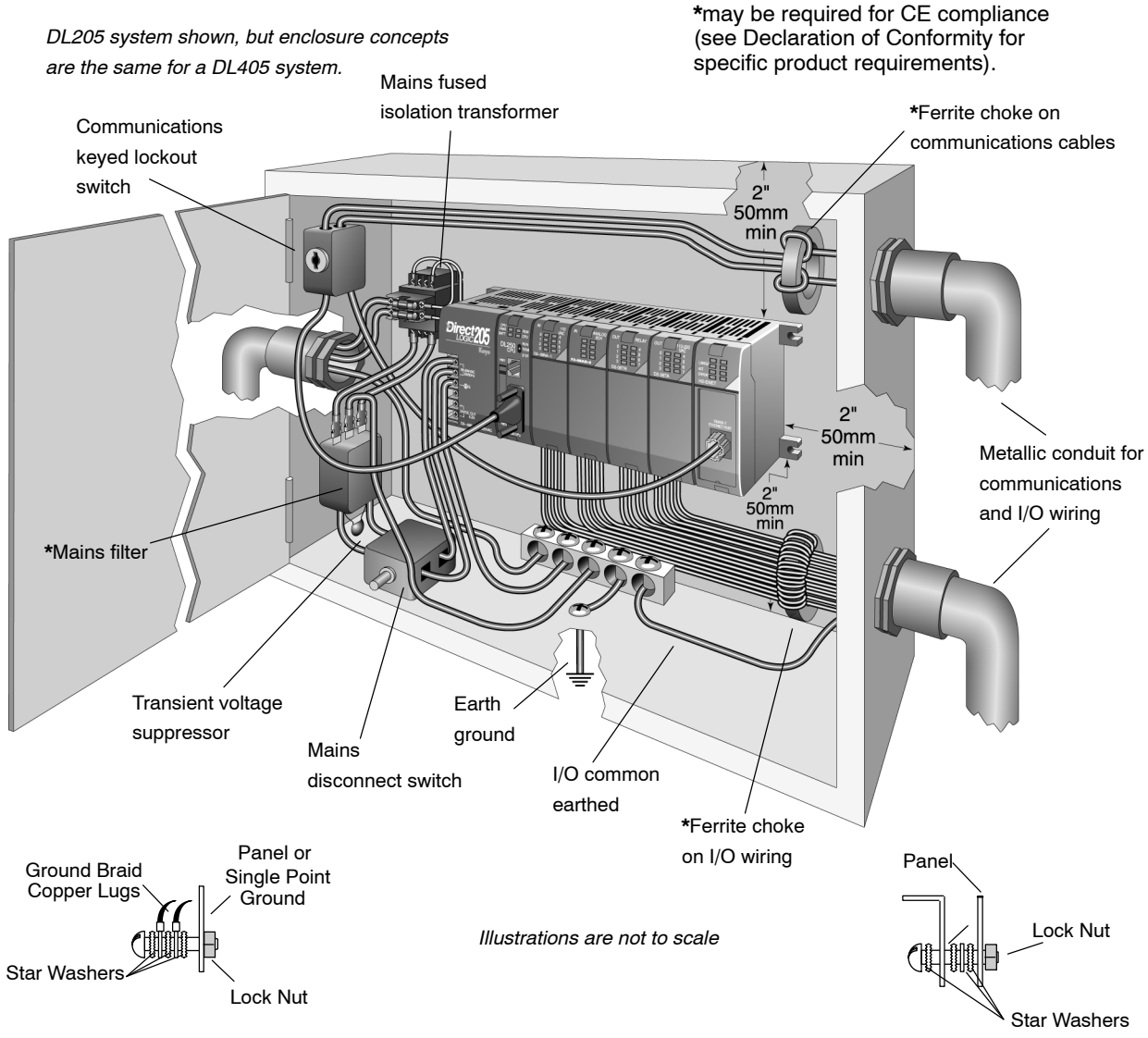
~~United Kingdom: the quickest contact is via the internet at~~
~~http://www.bsi.org.uk~~

See Errata Sheet at the beginning of this file. Replace the text in this box with the updated website urls and information provided.

Basic EMC Installation Guidelines

Enclosures

The simplest way to meet the safety requirements of the Machinery and Low Voltage Directives is to house all control equipment in an industry standard lockable steel enclosure. This normally has an added benefit because it will also help ensure that the EMC characteristics are well within the requirements of the EMC Directive. Although the RF emissions from the PLC equipment, when measured in the open air, are below the EMC Directive limits, certain configurations can increase emission levels. Holes in the enclosure, for the passage of cables or to mount operator interfaces, will often increase emissions.

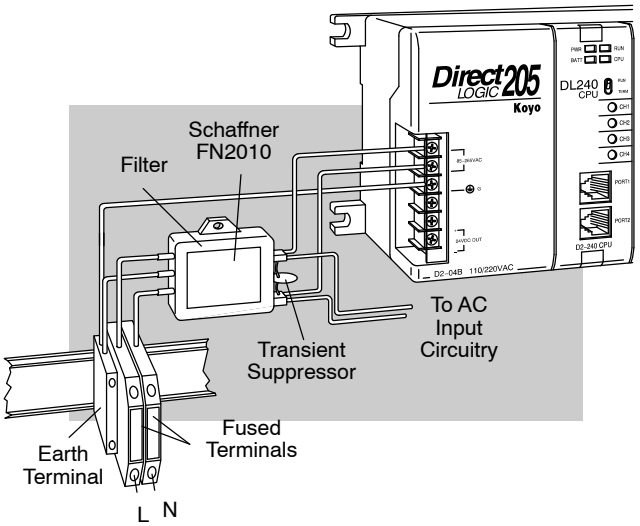


Electrostatic Discharge (ESD)

We specify in all declarations of conformity that our products are installed inside an industrial enclosure using metallic conduit for external wire runs; therefore, we test the products in a typical enclosure. However, we would like to point out that although our products operate normally in the presence of ESD, this is only the case when mounted within an enclosed industrial control cabinet. When the cabinet is open during installation or maintenance, the equipment and or programs may be at risk of damage from ESD carried by personnel.
We therefore recommend that all personnel take necessary precautions to avoid the risk of transferring static electricity to components inside the control cabinet. If necessary, clear warnings and instructions should be provided on the cabinet exterior, such as recommending the use of earth straps or similar devices, or the powering off of equipment inside the enclosure.

AC Mains Filters

The DL205 and DL305 AC powered base power supplies require extra mains filtering to comply with the EMC Directive on conducted RF emissions. All PLC equipment has been tested with filters from Schaffner, which reduce emissions levels if the filters are properly grounded (earth ground). A filter with a current rating suitable to supply all PLC power supplies and AC input modules should be selected. We suggest the FN2010 for the DL205 systems and the FN2080 for DL305 systems. The DL05, DL06 and DL405 systems do not require extra filtering.



NOTE: Very few mains filters can reduce problem emissions to negligible levels. In some cases, filters may increase conducted emissions if not properly matched to the problem emissions.

Suppression and Fusing

In order to comply with the fire risk requirements of the Low Voltage and Machinery Directive electrical standards EN 61010–1, and EN 60204–1, by limiting the power into “unlimited” mains circuits with power leads reversed, it is necessary to fuse both AC and DC supply inputs. You should also install a transient voltage suppressor across the power input connections of the PLC. Choose a suppressor such as a metal oxide varistor, with a rating of 275VAC working voltage for 230V nominal supplies (150VAC working voltage for 115V supplies) and high energy capacity (eg. 140 joules).

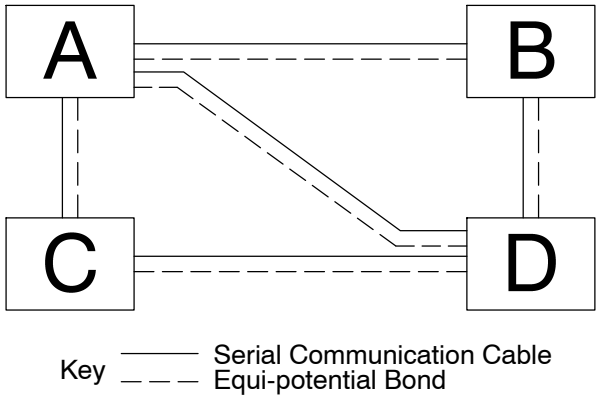
Transient suppressors must be protected by fuses and the capacity of the transient suppressor must be greater than the blow characteristics of the fuses or circuit breakers to avoid a fire risk. A recommended AC supply input arrangement for Koyo PLCs is to use twin 3 amp TT fused terminals with fuse blown indication, such as DINnectors DN–F10L terminals, or twin circuit breakers, wired to a Schaffner FN2010 filter or equivalent, with high energy transient suppressor soldered directly across the

output terminals of the filter. PLC system inputs should also be protected from voltage impulses by deriving their power from the same fused, filtered, and surge-suppressed supply.

Internal Enclosure Grounding

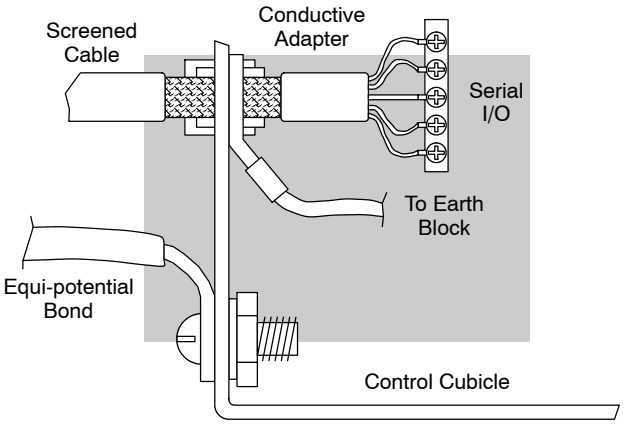
A heavy-duty star earth terminal block should be provided in every cubicle for the connection of all earth ground straps, protective earth ground connections, mains filter earth ground wires, and mechanical assembly earth ground connections. This should be installed to comply with safety and EMC requirements, local standards, and the requirements found in IEC 1000–5–2.The Machinery Directive also requires that the common terminals of PLC input modules, and common supply side of loads driven from PLC output modules should be connected to the protective earth ground terminal.

Equi–potential Grounding



Adequate site earth grounding must be provided for equipment containing modern electronic circuitry. The use of isolated earth electrodes for electronic systems is forbidden in some countries. Make sure you check any requirements for your particular destination. IEC 1000–5–2 covers equi-potential bonding of earth grids adequately, but special attention should be given to apparatus and control cubicles that contain I/O devices, remote I/O racks, or have inter-system communications with the primary PLC system enclosure. An equi-potential bond wire must be provided alongside all serial communications cables, and to any separate items of the plant which contain I/O devices connected to the PLC. The diagram shows an example of four physical locations connected by a communications cable.

Communications and Shielded Cables



Good quality 24 AWG minimum twisted-pair shielded cables, with overall foil and braid shields are recommended for analog cabling and communications cabling outside of the PLC enclosure. To date, it has been a common practice to only provide an earth ground for one end of the cable shield in order to minimize the risk of noise caused by earth ground loop currents between apparatus. The procedure of only grounding one end, which primarily originated as a result of trying to reduce hum in audio systems, is no longer applicable to the complex industrial environment. Shielded cables are also efficient emitters of RF noise from the PLC system, and can interact in a parasitic manner in networks and between multiple sources of interference.

The recommendation is to use shielded cables as electrostatic “pipes” between apparatus and systems, and to run heavy gauge equi-potential bond wires alongside all shielded cables. When a shielded cable runs through the metallic wall of an enclosure or machine, it is recommended in IEC 1000-5-2 that the shield should be connected over its full perimeter to the wall, preferably using a conducting adapter, and not via a pigtail wire connection to an earth ground bolt. Shields must be connected to every enclosure wall or machine cover that they pass through.

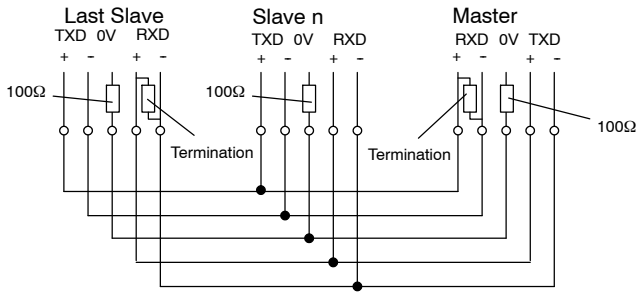
NOTE: Cables, whether shielded or not, connecting to the following modules MUST be enclosed within earthed metal conduit or other metallic trunking when outside the PLC enclosure: H4-EBC, H4-ECOM, F4-CPxxx-x and F4-SDS.

Analog and RS232 Cables

Providing an earth ground for both ends of the shield for analog circuits provides the perfect electrical environment for the twisted pair cable as the loop consists of signal and return, in a perfectly balanced circuit arrangement, with connection to the common of the input circuitry made at the module terminals. RS232 cables are handled in the same way.

Multidrop Cables

RS422 twin twisted pair, and RS485 single twisted pair cables also require a 0V link, which has often been provided in the past by the cable shield. It is now recommended that you use triple twisted pair cabling for RS422 links, and twin twisted pair cable for RS485 links. This is because the extra pair can be used as the 0V inter-system link. With loop DC power supplies earth grounded in both systems, earth loops are created in this manner via the inter-system 0v link. The installation guides encourage earth loops, which are maintained at a low impedance by using heavy equi-potential bond wires. **To account for non-European installations using single-end earth grounds, and sites with far from ideal earth ground characteristics, we recommend the addition of 100 ohm resistors at each 0V link connection in network and communications cables.**



Shielded Cables within Enclosures

When you run cables between PLC items within an enclosure which also contains susceptible electronic equipment from other manufacturers, remember that these cables may be a source of RF emissions. There are ways to minimize this risk. Standard data cables connecting PLCs and/or operator interfaces should be routed well away from other equipment and their associated cabling. You can make special serial cables where the cable shield is connected to the enclosure's earth ground at both ends, the same way as external cables are connected.

Additional Caution about RF Interference to Analog Modules

The readings from all analog modules can be affected by the use of devices that exhibit high field strengths such as mobile phones and motor drives.

All AutomationDirect products are tested to withstand field strength levels up to 10V/m. which is the maximum required by the relevant EU standards. While all products pass this test, analog modules will typically exhibit deviations of their readings. This is quite normal, however, systems designers should be aware of this and plan accordingly.

When assembling a control system using analog modules, these issues must be adhered to and should be integrated into the system design. This is the responsibility of the system builder/commissioner.

Network Isolation

For safety reasons, it is a specific requirement of the Machinery Directive that a keyswitch must be provided that isolates any network input signal during maintenance, so that remote commands cannot be received that could result in the operation of the machinery. The FA-ISOCON does not have a keyswitch! Use a keylock and switch on your enclosure which when open removes power from the FA-ISOCON. To avoid the introduction of noise into the system, any keyswitch assembly should be housed in its own earth grounded steel box and the integrity of the shielded cable must be maintained.

Again, for further information on EU directives we recommend that you get a copy of our EU Installation Manual (DA-EU-M). Also, if you connect to the World Wide Web, you can check the EU Commission's official website at: http://ec.europa.eu/index_en.htm.

DC Powered Versions

Due to slightly higher emissions radiated by the DC powered versions of the DL405, and the differing emissions performance for the different DC supply voltages, the following stipulations must be met:

The PLC must be housed within a metallic enclosure with a minimum amount of orifices.

I/O and communications cabling existing in the cabinet must be contained within metallic conduit/trunking.

Items Specific to the DL405

This equipment must be properly installed ~~while adhering to the guidelines of the PLC installation manual DA-EU-M~~, and is suitable for EN 61010-1 installation categories 1 or 2.

See Errata Sheet at the beginning of this file. Manual DA-EU-M is no longer available. Refer to the websites listed on the Errata Sheet for installation information.

The rating between all circuits in this product are rated as **basic insulation only**, as appropriate for single fault conditions.

The protection provided by the equipment may be impaired if the equipment is used in a manner not specified by the manufacturer.

It is the responsibility of the system designer to earth one side of all control and power circuits, and to earth the braid of screened cables.

Input power cables must be externally fused and have an externally mounted switch or circuit breaker, preferably mounted near the PLC.

NOTE: The DL405 internal base power supply has a 2A@250V slow blow fuse; however, it is not replaceable, so external fusing is required.

When needed, carefully clean the outside plastic case of PLC components using a dry cloth.

For hardware maintenance instructions, see the Maintenance and Troubleshooting section in this manual. This section also includes battery replacement information. Also, only replacement parts supplied by ***Automationdirect.com*** or its agents should be used.

Index

A

Agency Approvals, 2–7

ASCII Table, G–2

Auxiliary Functions, 3–16, A–2
- Handheld Programmer Configuration, A–10
- Memory Cartridge Operations, A–11
- Password Operations, A–12
- RLL Operations, A–5

B

Bases, 2–10
- CPU DIP Switches, 2–12
- Types, 2–11

Battery Backup, 3–12

BCD Numbers, 3–33

Bit Maps, 3–44 to 3–57

Boolean Instructions, 5–4, 5–10 to 5–25

C

Cables, Programming, 2–15

Clock and Calendar, A–8
- Instructions, 5–175

Communication Ports, Specifications, 3–9 to 3–12

Connect Solid State Devices, 2–21

Connector, I/O Ribbon Cable, 2–31

CPU
- Auxiliary Functions, A–2
- Battery Replacement, 9–2
- Clock and Calendar, 3–15
- Features, 3–2, 3–6
- Memory Map, 3–34
- Operating Modes, A–4
- Operating System, 3–18
- Password Protection, 3–15
- Scan Time, 3–26
- Setup, A–8
  - Set the Network Address, A–9
- Specifications, 3–4 to 3–6
- Status Indicators, 9–9

CPU Control Instructions, 5–177

D

Diagnostics, 3–23

DIP Switches, 3–8

DirectNET, Port Configuration, 4–21

Disabling Outputs, A–9

DL405 Aliases, 3–43

Drum Instruction Programming, 6–2
- Control Techniques, 6–10
- Drum Operation, 6–8
- Step Transitions, 6–4 to 6–7

Drum Instructions, 6-12 to 6-14
Drum instructions, EDRUM, 6-14

E

Error Codes, B-2 to B-8
European Directives, I-2
Event drum, 6-15

F

Force I/O, 3-20

H

Hexadecimal Numbers, 3-33

I

I/O Modules, I/O Configuration, A-6
I/O Response Time, 3-24
I/O Wiring and Specifications, 2-27
  I/O Module Charts, 2-33
Instruction Execution Times, C-2 to C-22
Instructions
  Accumulator/Data Stack, 5-54 to 5-57
  ASCII to HEX, 5-135
  BCD to Binary, 5-128
  Binary to BCD, 5-129
  Binary to Real, 5-132
  Bit Operations, 5-121 to 5-127
  Comparative Boolean, 5-26 to 5-31
  Counters, 5-46 to 5-48
  Encode, 5-126
  End Statement, 5-4
  Execution Times, C-2 to C-28
  Gray Code, 5-139
  HEX to ASCII, 5-136
  Immediate Instructions, 5-32 to 5-40
  Interrupts, 5-186
  Load Label, 5-162
  Master Control Relays, 5-183
  Move, 5-146
  Move Memory Cartridge, 5-162
  Out, 5-65 to 5-68
  Real to Binary, 5-133
  Set Bit, 5-167
  Shift Register, 5-53
  Stage, 7-23
  Stage Counter, 5-49 to 5-50
  Stage Programming, 7-2
  Subroutines, 5-181
  Timers, 5-41 to 5-45
  Up/Down Counter, 5-51 to 5-52
Intelligent I/O Instructions, 5-189
Interrupt Instructions, 5-185
Instructions, Compare, 5-82 to 5-86

K

Keyswitch Functions, 3-7

L

Local I/O Expansion, 4-11

M

Maintenance, 9-2
  Battery Maintenance, 9-2
  Code Identifiers, 9-6
  Diagnostics, 9-4
  Error Code Locations, 9-5
  System Error Codes, 9-8
  Test Modes, 9-19
Manual Organization, 1-2, 1-6
Masked Drums, 6-18
Math Instructions, 5-87
  Increment, 5-117
  Increment Binary, 5-120
  Transcendental Functions, 5-118
Memory, F-2
  Set Retentive Ranges, A-9
Memory Cartridge, 3-14
Memory cartridge, battery replacement, 9-3
Memory Map
  DL430, 3-40
  DL440, 3-41
  DL450, 3-42
Message Instructions, 5-194 to 5-204
MODBUS, Port Configuration, 4-20
Module Placement, 4-4
  I/O Configuration, 4-5

Mounting Guidelines, 2-6
Base Dimensions, 2-5
Panel Layout, 2-6

N

Network Address, A-9
Network connections, MODBUS & ***Direct***NET, 4-18
Network Instructions, 5-191
Network Master Operation, 4-28 to 4-31
Network Slave Operation, 4-22
Number Conversion Instructions, 5-128
Numbering Systems
BCD, H-5
Binary, H-2
Floating Point, H-6
Hexadecimal, H-3
Octal, H-4

O

Octal Numbers, 3-32

P

Part Numbering, 1-8
Passwords, A-12
PID
Analog Filter, 8-53
Bumpless Transfer, 8-13, 8-27
Cascade Control, 8-63
Tuning, 8-67
Control Introduction, 8-4
DirectSOFT 5 Filter, 8-54
DL450 Control, 8-6
Error Flags, 8-18
Error Term Selection, 8-33
Loop Modes, 8-28, 8-52
On/Off Control, 8-68
Operation, 8-9
Parameters, 8-32
PID View, 8-49 to 8-51
Ramp/Soak, 8-39
Reset Windup, 8-10, 8-34
Special Features, 8-52
Time Proportioning, 8-68
PID Alarms
Alarm Features, 8-3
Auto Tuning Error, 8-47
Hysteresis, 8-13, 8-36, 8-38
Monitor Limit, 8-35
Overflow/Underflow Error, 8-38
Programming Error, 8-38
PV Deviation, 8-36
Rate-of-Change, 8-13, 8-37
Setup Alarms, 8-35
PID Loop
Alarms, 8-13
Configure, 8-26
Features, 8-2, 8-3
Feedforward Control, 8-70
Freeze Bias, 8-11, 8-34
Loop Definitions, 8-21
Mode, 8-28
Operating Modes, 8-14
Special Loop Calculations, 8-14
Setup, 8-18
Terminology, 8-74
Time-Proportioning Control, On/Off Control Example, 8-67
Transfer Mode, 8-27
Troubleshooting Tips, 8-72, 8-74
Tuning, 8-40
Auto , 8-44
Manual, 8-41 to 8-44
PID Mode 2 Word Description, 8-23
PID Mode Setting 1 Description, 8-22
PID Position Algorithm, 8-9, 8-15
Position Form, 8-9
PID Velocity Algorithm, 8-9
Algorithm Form, 8-12
Velocity Algorithm, 8-15
Pointers, 5-58
Power Budget, 4-7
Calculation Example, 4-9
Power Budget , Worksheet, 4-10
Product Weights, E-2
Program Control Instructions, 5-179
Program Media, 3-13
Program Mode, 3-19
Programming
Accumulator Logic Instructions, 5-70 to 5-85
Bit Operation Instructions, 5-121 to 5-127
Comparative Boolean, 5-7
Compare Instructions, 5-82 to 5-86
Counters, 5-46 to 5-48
Instruction Set Table, 5-2 to 5-4
Load Instructions, 5-59 to 5-64

Midline Outputs, 5-5
Out Instructions, 5-65 to 5-68
Pointers, 5-58
POP Instruction, 5-69
Shift Register, 5-53
Timers, 5-41 to 5-45

Programming Methods, 1-4

Q

Quick Start, 1-10, 1-12

R

Ramp/Soak Generator, 8-56
Controls, 8-58
DirectSOFT Example, 8-60
Flag Bit Description Table, 8-24
Profile Monitoring, 8-59
Table, 8-56
Table Flags, 8-58
Table Location, 8-25
Test the Profile, 8-62, 8-64
Testing, 8-59

Remote I/O Expansion, 4-12

Retentive Memory, A-9

Run Mode, 3-19

S

Safety Guidelines, 2-2
Emergency Stops, 2-3

Safety guidelines, Orderly System Shutdown, 2-4

Scan Time
Display, A-8
Set the Watchdog Timer, A-8

Scan Time Configuration, 3-15

Sinking/Sourcing, 2-19

Special Relays, D-2 to D-4

Specialty I/O, 3-20

Stage Programming, 7-2
Garage Door Opener Example, 7-10
Initial Stages, 7-5
Introduction, 7-2
Jump Instruction, 7-7
Parallel Processing Concepts, 7-19
Power Flow Transition, 7-18
Program Organization, 7-15
Stage View, 7-28
State Transition Diagrams, 7-3
Unconditional Outputs, 7-18

Status Indicators, 3-7

System Components, 1-4

System CRs, 3-39

System Design, 1-16

System Design Strategies, 4-2

System Parameters, 3-38

T

Table Instructions, 5-142

Technical Support, 1-2, 1-6

Test Modes, A-9

Timed Drum, 6-16

Transient Suppression, 2-23

Troubleshooting
Electrical Noise, 9-16
I/O modules, 9-14

V

V-memory Operations, A-6

W

Wiring
Expansion Unit, 2-14
I/O Module, 2-30
I/O Strategies, 2-16
I/O wiring strategies, 2-17
Multi-point Modules, 2-31
Power Input, 3-8

Wiring Guidelines, 2-13