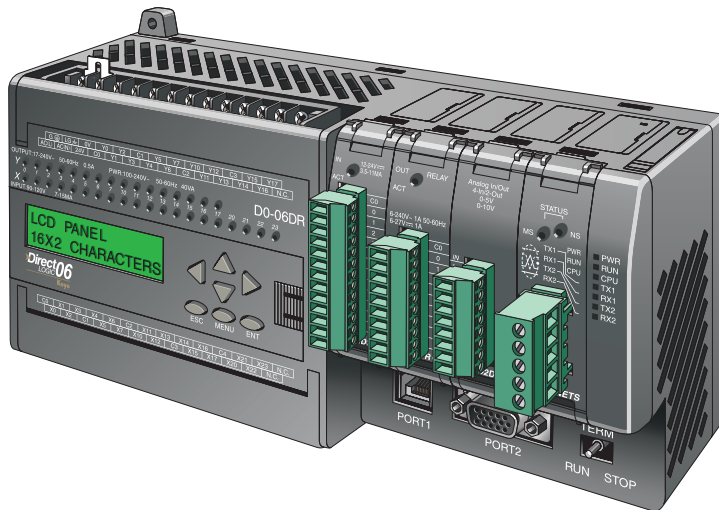




# DL06 User Manual

Manual Number: D0-06USER-M

Volume 1 of 2



# DL06 MICRO PLC USER MANUAL

---



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

**Manual Number:** D0-06USER-M  
**Issue:** First Edition, Rev. A  
**Issue Date:** 10/02

Publication History		
Issue	Date	Description of Changes
First Edition	7/02	Original
Rev. A	10/02	Updated drawing images and made minor corrections.

# VOLUME ONE:

## TABLE OF CONTENTS

---



### Chapter 1: Getting Started

<b>Introduction</b>	<b>1-2</b>
The Purpose of this Manual	1-2
Supplemental Manuals	1-2
Technical Support	1-2
<b>Conventions Used</b>	<b>1-3</b>
Key Topics for Each Chapter	1-3
<b>DL06 Micro PLC Overview</b>	<b>1-4</b>
The DL06 PLC Features	1-4
<b>Programming Methods</b>	<b>1-4</b>
DirectSOFT32 Programming for Windows™	1-4
Handheld Programmer	1-5
<b>I/O Quick Selection Guide</b>	<b>1-5</b>
<b>Quick Start</b>	<b>1-6</b>
<b>Steps to Designing a Successful System</b>	<b>1-10</b>
<b>Questions and Answers about DL06 Micro PLCs</b>	<b>1-12</b>

### Chapter 2: Installation, Wiring, and Specifications . . . . .2-1

<b>Safety Guidelines</b>	<b>2-2</b>
Plan for Safety	2-2
Three Levels of Protection	2-2
Orderly System Shutdown	2-3
System Power Disconnect	2-3
Emergency Stop	2-3
Class I, Division 2 Approval	2-4
<b>Orientation to DL06 Front Panel</b>	<b>2-4</b>

## Table of Contents

---

Terminal Block Removal .....	2-5
<b>Mounting Guidelines .....</b>	<b>2-6</b>
Unit Dimensions .....	2-6
Enclosures .....	2-6
Panel Layout & Clearances .....	2-7
Using Mounting Rails .....	2-8
Environmental Specifications .....	2-9
Agency Approvals .....	2-9
<b>Wiring Guidelines .....</b>	<b>2-10</b>
Fuse Protection for Input Power .....	2-10
External Power Source .....	2-11
Planning the Wiring Routes .....	2-11
Fuse Protection for Input and Output Circuits .....	2-12
I/O Point Numbering .....	2-12
<b>System Wiring Strategies .....</b>	<b>2-13</b>
PLC Isolation Boundaries .....	2-13
Connecting Operator Interface Devices .....	2-14
Connecting Programming Devices .....	2-14
Sinking / Sourcing Concepts .....	2-15
I/O “Common” Terminal Concepts .....	2-16
Connecting DC I/O to “Solid State” Field Devices .....	2-17
Solid State Input Sensors .....	2-17
Solid State Output Loads .....	2-17
Relay Output Wiring Methods .....	2-19
Surge Suppression For Inductive Loads .....	2-20
Prolonging Relay Contact Life .....	2-21
DC Input Wiring Methods .....	2-22
DC Output Wiring Methods .....	2-23
High-Speed I/O Wiring Methods .....	2-24
<b>Glossary of Specification Terms .....</b>	<b>2-25</b>
<b>Wiring Diagrams and Specifications .....</b>	<b>2-26</b>
D0-06AA I/O Wiring Diagram .....	2-26
D0-06AR I/O Wiring Diagram .....	2-28
D0-06DA I/O Wiring Diagram .....	2-30
D0-06DD1 I/O Wiring Diagram .....	2-32
D0-06DD2 I/O Wiring Diagram .....	2-34
D0-06DR I/O Wiring Diagram .....	2-36
D0-06DD1-D I/O Wiring Diagram .....	2-38

D0-06DR-D I/O Wiring Diagram .....	2-40
Discrete Options Modules .....	2-42

## Chapter 3: High-speed Input and Pulse Output Features ..... 3-1

<b>Introduction</b> .....	3-2
Built-in Motion Control Solution .....	3-2
Availability of HSIO Features .....	3-2
Dedicated High-Speed I/O Circuit .....	3-3
Wiring Diagrams for Each HSIO Mode .....	3-3

<b>Choosing the HSIO Operating Mode</b> .....	3-4
Understanding the Six Modes .....	3-4
Default Mode .....	3-5
Configuring the HSIO Mode .....	3-6
Configuring Inputs X0 – X3 .....	3-6

<b>Mode 10: High-Speed Counter</b> .....	3-7
Purpose .....	3-7
Functional Block Diagram .....	3-7
Wiring Diagram .....	3-8
Interfacing to Counter Inputs .....	3-8
Setup for Mode 10 .....	3-9
Presets and Special Relays .....	3-9
Absolute and Incremental Presets .....	3-10
Preset Data Starting Location .....	3-11
Using Fewer than 24 Presets .....	3-11
Equal Relay Numbers .....	3-12
Calculating Your Preset Values .....	3-13
X Input Configuration .....	3-14
Writing Your Control Program .....	3-15
Program Example 1: Counter Without Presets .....	3-16
Program Example Cont'd .....	3-17
Program Example 2: Counter With Presets .....	3-18
Program Example 3: Counter With Preload .....	3-21
Troubleshooting Guide for Mode 10 .....	3-23
Symptom: The counter does not count. ....	3-23
Symptom: The counter counts but the presets do not function. ....	3-23
Symptom: The counter counts up but will not reset. ....	3-23

<b>Mode 20: Up/Down Counter</b> .....	<b>3-24</b>
Purpose .....	3-24
Functional Block Diagram .....	3-24
Quadrature Encoder Signals .....	3-25
Wiring Diagram .....	3-25
Interfacing to Encoder Outputs .....	3-26
Setup for Mode 20 .....	3-27
Presets and Special Relays .....	3-27
X Input Configuration .....	3-28
Mode 20 Up/Down Counter .....	3-28
Writing Your Control Program .....	3-29
Program Example 1 Quadrature Counting with an Interrupt .....	3-30
Program Example: 2 Up/Down Counting with Standard Inputs .....	3-32
Program Example: 3 Quadrature Counting .....	3-34
Troubleshooting Guide for Mode 20 .....	3-37
Symptom: The counter does not count. ....	3-37
Symptom: The counter counts in the wrong direction .....	3-37
Symptom: The counter counts up and down but will not reset. ....	3-37
<b>Mode 30: Pulse Output</b> .....	<b>3-38</b>
Purpose .....	3-38
Functional Block Diagram .....	3-39
Wiring Diagram .....	3-40
Interfacing to Drive Inputs .....	3-40
Motion Profile Specifications .....	3-41
Physical I/O Configuration .....	3-41
Logical I/O Functions .....	3-41
Setup for Mode 30 .....	3-42
Profile / Velocity Select Register .....	3-43
Profile Parameter Table .....	3-43
Automatic Trapezoidal Profile .....	3-43
Step Trapezoidal Profile .....	3-44
Choosing the Profile Type .....	3-45
Automatic Trapezoidal Profile Defined .....	3-45
Step Trapezoidal Profiles Defined .....	3-46
Velocity Control Defined .....	3-46
Automatic Trapezoidal Profile Operation .....	3-47
Program Example 1: Automatic Trapezoidal Profile .....	3-48

Preload Position Value .....	3-49
Program Example 2: Automatic Trapezoidal Profile .....	3-50
Program Example 3: Home Search Automatic Trapezoidal Profile .....	3-53
Step Trapezoidal Profile Operation .....	3-55
Program Example 4: Step Trapezoidal Profile .....	3-56
Velocity Profile Operation .....	3-59
Program Example 5: Velocity Profile .....	3-60
Automatic Trapezoidal Profile Error Codes .....	3-62
Troubleshooting Guide for Mode 30 .....	3-62
Symptom: The stepper motor does not rotate. ....	3-62
Symptom: The motor turns in the wrong direction. ....	3-63
<b>Mode 40: High-Speed Interrupts .....</b>	<b>3-64</b>
Purpose .....	3-64
Functional Block Diagram .....	3-64
Setup for Mode 40 .....	3-65
Interrupts and the Ladder Program .....	3-65
External Interrupt Timing Parameters .....	3-66
Timed Interrupt Parameters .....	3-66
X Input / Timed INT Configuration .....	3-66
Program Example 1: External Interrupt .....	3-67
Program Example 2: Timed Interrupt .....	3-68
<b>Mode 50: Pulse Catch Input .....</b>	<b>3-69</b>
Purpose .....	3-69
Functional Block Diagram .....	3-69
Pulse Catch Timing Parameters .....	3-69
When to use Pulse Catch Mode .....	3-70
Setup for Mode 50 .....	3-70
X Input Configuration .....	3-71
Program Example 1: Pulse Catch .....	3-72
<b>Mode 60: Discrete Inputs with Filter .....</b>	<b>3-73</b>
Purpose .....	3-73
Functional Block Diagram .....	3-73
Input Filter Timing Parameters .....	3-73
Setup for Mode 60 .....	3-74
X Input Configuration .....	3-74
Program Example: Filtered Inputs .....	3-75

<b>Chapter 4: CPU Specifications and Operation</b> .....	<b>4-1</b>
<b>Introduction</b> .....	<b>4-2</b>
DL06 CPU Features .....	4-2
<b>CPU Specifications</b> .....	<b>4-3</b>
<b>CPU Hardware Setup</b> .....	<b>4-4</b>
Communication Port Pinout Diagrams .....	4-4
Connecting the Programming Devices .....	4-5
CPU Setup Information .....	4-5
Status Indicators .....	4-6
Mode Switch Functions .....	4-6
Changing Modes in the DL06 PLC .....	4-7
Mode of Operation at Power-up .....	4-7
<b>Using Battery Backup</b> .....	<b>4-8</b>
Enabling the Battery Backup .....	4-8
Auxiliary Functions .....	4-9
Clearing an Existing Program .....	4-9
Initializing System Memory .....	4-9
Setting Retentive Memory Ranges .....	4-10
Using a Password .....	4-11
<b>CPU Operation</b> .....	<b>4-12</b>
CPU Operating System .....	4-12
Program Mode .....	4-13
Run Mode .....	4-13
Read Inputs .....	4-14
Service Peripherals and Force I/O .....	4-14
CPU Bus Communication .....	4-15
Update Clock, Special Relays and Special Registers .....	4-15
Solve Application Program .....	4-16
Solve PID Loop Equations .....	4-16
Write Outputs .....	4-17
Write Outputs to Specialty I/O .....	4-17
Diagnostics .....	4-17
<b>I/O Response Time</b> .....	<b>4-17</b>
Is Timing Important for Your Application? .....	4-17
Normal Minimum I/O Response .....	4-18
Normal Maximum I/O Response .....	4-18



Improving Response Time .....	4-19
<b>CPU Scan Time Considerations .....</b>	<b>4-20</b>
Reading Inputs .....	4-20
Writing Outputs .....	4-20
Service Peripherals .....	4-21
CPU Bus Communication .....	4-21
Update Clock / Calendar, Special Relays, Special Registers .....	4-21
Application Program Execution .....	4-22
PLC Numbering Systems .....	4-23
PLC Resources .....	4-23
V-Memory .....	4-24
Binary-Coded Decimal Numbers .....	4-24
Hexadecimal Numbers .....	4-24
<b>Memory Map .....</b>	<b>4-25</b>
Octal Numbering System .....	4-25
Discrete and Word Locations .....	4-25
V Memory Locations for Discrete Memory Areas .....	4-25
Input Points (X Data Type) .....	4-26
Output Points (Y Data Type) .....	4-26
Control Relays (C Data Type) .....	4-26
Timers and Timer Status Bits (T Data Type) .....	4-26
Timer Current Values (V Data Type) .....	4-27
Counters and Counter Status Bits (CT Data type) .....	4-27
Counter Current Values (V Data Type) .....	4-27
Word Memory (V Data Type) .....	4-28
Stages (S Data type) .....	4-28
Special Relays (SP Data Type) .....	4-28
<b>DL06 System V-memory .....</b>	<b>4-29</b>
System Parameters and Default Data Locations (V Data Type) .....	4-29
DL06 Memory Map .....	4-31
X Input / Y Output Bit Map .....	4-32
Stage Control / Status Bit Map .....	4-33
<b>Control Relay Bit Map .....</b>	<b>4-35</b>
<b>Timer Status Bit Map .....</b>	<b>4-37</b>
<b>Counter Status Bit Map .....</b>	<b>4-37</b>

<b>Remote I/O Bit Map</b>	<b>4-38</b>
<b>Module Placement</b>	<b>4-42</b>
Slot Numbering	4-42
Automatic I/O Configuration	4-43
Manual I/O Configuration	4-43
<b>Power Budgeting</b>	<b>4-44</b>
Power supplied	4-44
Power required by base unit	4-44
Power required by option cards	4-44
Configuring the DL06's Comm Ports	4-46
DL06 Port Specifications	4-46
DL06 Port Pinouts	4-46
Choosing a Network Specification	4-47
RS-232 Network	4-47
RS-485 Network	4-47
Connecting to MODBUS and DirectNET Networks	4-48
<b>MODBUS Port Configuration</b>	<b>4-48</b>
DirectNET Port Configuration	4-49
<b>Non-Sequence Protocol (ASCII In/Out and PRINT)</b>	<b>4-50</b>
MODBUS Port Configuration	4-50
<b>Network Slave Operation</b>	<b>4-51</b>
MODBUS Function Codes Supported	4-51
Determining the MODBUS Address	4-51
If Your Host Software Requires the Data Type and Address	4-52
Example 1: V2100	4-53
Example 2: Y20	4-53
Example 3: T10 Current Value	4-53
Example 4: C54	4-53
If Your MODBUS Host Software Requires an Address ONLY	4-54
Example 1: V2100 584/984 Mode	4-55
Example 2: Y20 584/984 Mode	4-55
Example 3: T10 Current Value 484 Mode	4-55
Example 4: C54 584/984 Mode	4-55
Determining the DirectNET Address	4-55
<b>Network Master Operation</b>	<b>4-56</b>
Step 1: Identify Master Port # and Slave #	4-57

Step 2: Load Number of Bytes to Transfer .....	4-57
Step 3: Specify Master Memory Area .....	4-58
Step 4: Specify Slave Memory Area .....	4-58
Communications from a Ladder Program .....	4-59
Multiple Read and Write Interlocks .....	4-59
<b>Network Master Operation (using MRX and MWX Instructions) .....</b>	<b>4-60</b>
MODBUS Function Codes Supported .....	4-60
MODBUS Port Configuration .....	4-61
MODBUS Read from Network(MRX) .....	4-62
MRX Slave Memory Address .....	4-63
MRX Master Memory Addresses .....	4-63
MRX Number of Elements .....	4-63
MRX Exception Response Buffer .....	4-63
MODBUS Write to Network (MWX) .....	4-64
MWX Slave Memory Address .....	4-65
MWX Master Memory Addresses .....	4-65
MWX Number of Elements .....	4-65
MWX Exception Response Buffer .....	4-65
MRX / MWX Example in DirectSOFT32 .....	4-66
Multiple Read and Write Interlocks .....	4-66
<b>Chapter 5: Standard RLL Instructions .....</b>	<b>5-1</b>
<b>Introduction .....</b>	<b>5-2</b>
<b>Using Boolean Instructions .....</b>	<b>5-5</b>
END Statement .....	5-5
Simple Rungs .....	5-5
Normally Closed Contact .....	5-5
Contacts in Series .....	5-6
Midline Outputs .....	5-6
Parallel Elements .....	5-6
Joining Series Branches in Parallel .....	5-7
Joining Parallel Branches in Series .....	5-7
Combination Networks .....	5-7
Comparative Boolean .....	5-7
Boolean Stack .....	5-8
Immediate Boolean .....	5-9

<b>Boolean Instructions</b> .....	<b>5-10</b>
<b>Comparative Boolean</b> .....	<b>5-26</b>
<b>Immediate Instructions</b> .....	<b>5-32</b>
<b>Timer, Counter and Shift Register Instructions</b> .....	<b>5-39</b>
Using Timers .....	5-39
Timer Example Using Discrete Status Bits .....	5-41
Timer Example Using Comparative Contacts .....	5-41
Accumulating Timer Example using Discrete Status Bits .....	5-43
Accumulator Timer Example Using Comparative Contacts .....	5-43
Using Counters .....	5-44
Counter Example Using Discrete Status Bits .....	5-46
Counter Example Using Comparative Contacts .....	5-46
Stage Counter Example Using Discrete Status Bits .....	5-48
Stage Counter Example Using Comparative Contacts .....	5-48
Up / Down Counter Example Using Discrete Status Bits .....	5-50
Up / Down Counter Example Using Comparative Contacts .....	5-50
<b>Accumulator / Stack Load and Output Data Instructions</b> .....	<b>5-52</b>
Using the Accumulator .....	5-52
Copying Data to the Accumulator .....	5-52
Changing the Accumulator Data .....	5-53
Using the Accumulator Stack .....	5-54
Using Pointers .....	5-55
<b>Logical Instructions (Accumulator)</b> .....	<b>5-69</b>
<b>Math Instructions</b> .....	<b>5-86</b>
<b>Transcendental Functions</b> .....	<b>5-118</b>
<b>Bit Operation Instructions</b> .....	<b>5-120</b>
<b>Number Conversion Instructions (Accumulator)</b> .....	<b>5-127</b>
Shuffle Digits Block Diagram .....	5-139
<b>Table Instructions</b> .....	<b>5-141</b>
Copy Data From a Data Label Area to V Memory .....	5-143
<b>Clock / Calendar Instructions</b> .....	<b>5-171</b>
<b>CPU Control Instructions</b> .....	<b>5-173</b>
<b>Program Control Instructions</b> .....	<b>5-175</b>
MLS/MLR Example .....	5-182

<b>Interrupt Instructions</b>	<b>.5-183</b>
Timed Interrupt Program Example	.5-185
Independent Timed Interrupt	.5-185
<b>Message Instructions</b>	<b>.5-186</b>
Fault Example	.5-186
Data Label Example	.5-188
Direct Text Entry	.5-197
Embedding date and/or time variables	.5-198
Embedding V-memory data	.5-198
Data Format Suffixes for Embedded V-memory Data	.5-199
Text Entry from V-memory	.5-200
<b>MODBUS RTU Instructions</b>	<b>.5-201</b>
MRX Slave Address Ranges	.5-202
MRX Example	.5-203
MWX Slave Address Ranges	.5-205
MWX Master Memory Address Ranges	.5-205
MWX Number of Elements	.5-205
MWX Exception Response Buffer	.5-205
MWX Example	.5-206
<b>ASCII Instructions</b>	<b>.5-207</b>
Reading ASCII Input Strings	.5-207
Writing ASCII Output Strings	.5-207
Managing the ASCII Strings	.5-208
AFIND Search Example	.5-214
AFIND Example Combined with AEX Instruction	.5-215

# GETTING STARTED

---



# CHAPTER 1

## In This Chapter...

Introduction .....	1-2
Conventions Used .....	1-3
DL06 Micro PLC Overview .....	1-4
Programming Methods .....	1-4
I/O Quick Selection Guide .....	1-5
Quick Start .....	1-6
Steps to Designing a Successful System .....	1-10
Questions and Answers about DL06 Micro PLCs .....	1-12

# Introduction

### The Purpose of this Manual

Thank you for purchasing a DL06 Micro PLC. This manual shows you how to install, program, and maintain all PLCs in the DL06 family. 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 DL06 PLCs and for the PLC programmer. This user manual will provide the information you need to get and keep your system up and running.

### Supplemental Manuals

The D0-OPTIONS-M manual contains technical information about the option cards available for the DL06 PLCs. This information includes specifications and wiring diagrams that will be indispensable if you use any of the optional I/O or communications cards. If you have purchased one of our operator interface panels or *DirectSOFT*™ programming software, you will want to refer to the manuals that are written for these products.

### Technical Support

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

**770-844-4200.**

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

**In Brazil: <http://www.soliton.com.br>**

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

# Conventions Used



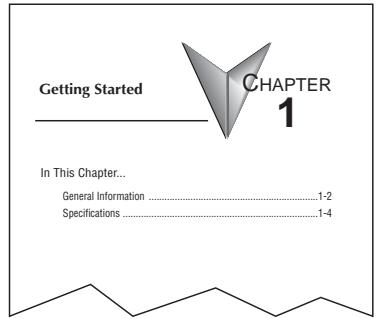
When you see the “notepad” icon in the left-hand margin, the paragraph to its immediate right will be a **special note**. Notes represent information that may make your work quicker or more efficient. The word **NOTE:** in boldface will mark the beginning of the text.



When you see the “exclamation point” icon in the left-hand margin, the paragraph to its immediate right will be a **warning**. This information could prevent injury, loss of property, or even death in extreme cases. Any warning in this manual should be regarded as critical information that should be read in its entirety. The word **WARNING** in boldface will mark the beginning of the text.

## Key Topics for Each Chapter

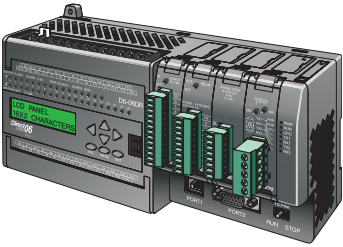
The beginning of each chapter will list the key topics that can be found in that chapter.





# DL06 Micro PLC Overview

The DL06 micro PLC family is a versatile product line that combines powerful features and a very compact footprint. The DL06 PLCs offer expandable I/O, high-speed counter, floating point, PID, etc. There are a number of communication options and an optional LCD display.



## The DL06 PLC Features

The DL06 Micro PLC family includes eight different versions. All have the same appearance and CPU performance. The CPU offers an instruction set very similar to our powerful new DL260 CPU including new easy to use ASCII and MODBUS instructions. All DL06 PLCs have two built-in communications ports that can be used for programming, operator interface, networking, etc.

Units with DC inputs have selectable high-speed input features on four input points. Units with DC outputs offer selectable pulse output capability on the first and second output points. Details of these features and more are covered in Chapter 4, CPU Specifications and Operation. There are eight versions of the DL06 PLC. The most common industrial I/O types and power supply voltages are available. Consult the following table to find the model number of the PLC that best fits your application.

DL06 Micro PLC Family					
DL06 Part Number	Discrete Input Type	Discrete Output Type	External Power	High-Speed Input	Pulse Output
D0-06AA	AC	AC	95-240 VAC	No	No
D0-06AR	AC	Relay	95-240 VAC	No	No
D0-06DA	DC	AC	95-240 VAC	Yes	No
D0-06DD1	DC	DC Sinking	95-240 VAC	Yes	Yes
D0-06DD2	DC	DC Sourcing	95-240 VAC	Yes	Yes
D0-06DR	DC	Relay	95-240 VAC	Yes	No
D0-06DD1-D	DC	DC Sinking	12-24 VDC	Yes	Yes
D0-06DR-D	DC	Relay	12-24 VDC	Yes	No

## Programming Methods

Two programming methods are available: RLL (Relay Ladder Logic) and RLL<sup>PLUS</sup>. RLL<sup>PLUS</sup> combines the added feature of flow chart programming (Stage) to the standard RLL language. Both the *DirectSOFT*<sup>TM</sup> programming package and the handheld programmer support RLL<sup>PLUS</sup> as well as standard RLL instructions.

### *DirectSOFT*32 Programming for Windows<sup>TM</sup>

The DL06 Micro PLC can be programmed with *DirectSOFT*32, V4.0 or later, a Windows-based software package that supports familiar features such as cut-and-paste between applications, point-and-click editing, viewing and editing multiple application programs at the same time, etc.

*Direct*SOFT32 (part number PC-PGMSW) supports the *Direct*LOGIC CPU families. You can use the full version of *Direct*SOFT32 to program the DL05, DL06, DL105, DL205, DL305, and DL405. (Upgrade software may be required for new CPUs when they become available). A separate manual discusses *Direct*SOFT32 programming software.

*Direct*SOFT32 version 4.0 or later is needed to program the DL06.

## Handheld Programmer

All DL06 Micro PLCs have a built-in programming port for use with the handheld programmer (D2–HPP), the same programmer used with the DL05, DL105 and DL205 families. The handheld programmer can be used to create, modify and debug your application program. A separate manual discusses the Handheld Programmer. Only D2–HPPs with firmware version 2.2 or later will program the DL06.

## I/O Quick Selection Guide

The eight versions of the DL06 have input/output circuits which can interface to a wide variety of field devices. In several instances a particular input or output circuit can interface to either DC or AC voltages, or both sinking and sourcing circuit arrangements. Check this guide to find the proper DL06 Micro PLC to interface to the field devices in your application.

I/O Selection Guide						
DL06 Part Number	INPUTS			OUTPUTS		
	I/O type/commons	Sink/Source	Voltage Ranges	I/O type/commons	Sink/Source	Voltage/ Current Ratings*
D0-06AA	AC / 5	–	90 – 120 VAC	AC / 4	–	17 – 240 VAC, 47 – 63 Hz 0.5A
D0-06AR	AC / 5	–	90 – 120 VAC	Relay / 4	Sink or Source	6 – 27VDC, 2A 6 – 240 VAC, 2A
D0-06DA	DC / 5	Sink or Source	12 – 24 VDC	AC / 4	–	17 – 240 VAC, 47 – 63 Hz 0.5A
D0-06DD1	DC / 5	Sink or Source	12 – 24 VDC	DC / 4	Sink	6 – 27 VDC, 0.5A (Y0–Y1) 6 – 27 VDC, 1.0A (Y2–Y17)
D0-06DD2	DC / 5	Sink or Source	12 – 24 VDC	DC / 4	Source	6 – 27 VDC, 0.5A (Y0–Y1) 6 – 27 VDC, 1.0A (Y2–Y17)
D0-06DR	DC / 5	Sink or Source	12 – 24 VDC	Relay / 4	Sink or Source	6 – 27VDC, 2A 6 – 240 VAC, 2A
D0-06DD1–D	DC / 5	Sink or Source	12 – 24 VDC	DC / 4	Sink	6 – 27 VDC, 0.5A (Y0–Y1) 6 – 27 VDC, 1.0A (Y2–Y17)
D0-06DR–D	DC / 5	Sink or Source	12 – 24 VDC	Relay / 4	Sink or Source	6 – 27 VDC, 2A 6 – 240 VAC, 2A

\* See Chapter 2, Specifications for more information about a particular DL06 version.

# Quick Start

This example is not intended to tell you everything you need to know about programming and starting-up a complex control system. It is only intended to give you an opportunity to demonstrate to yourself and others the basic steps necessary to power up the PLC and confirm its operation. Please look for warnings and notes throughout this manual for important information you will not want to overlook.

### Step 1: Unpack the DL06 Equipment

Unpack the DL06 and gather the parts necessary to build this demonstration system. The recommended components are:

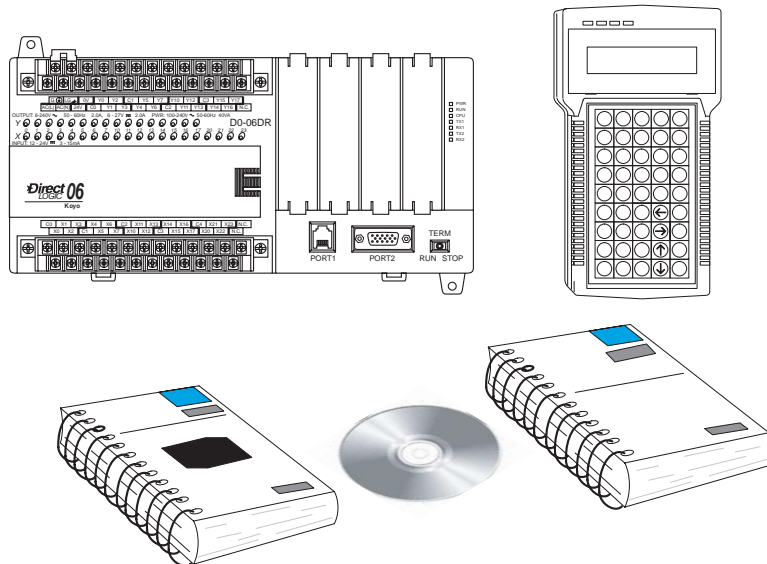
- DL06 Micro PLC
- AC power cord or DC power supply
- Toggle switches (see Step 2 on next page).
- Hook-up wire, 16-22 AWG
- DL06 User Manual (this manual)
- A small screwdriver, 5/8" flat or #1 Philips type

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

- *Direct*SOFT32 Programming Software V4.0 or later (PC-PGMSW or PC-PGM-BRICK), *Direct*SOFT32 Manual (included with the software), and a programming cable (D2-DSCBL connects the DL06 to a personal computer)

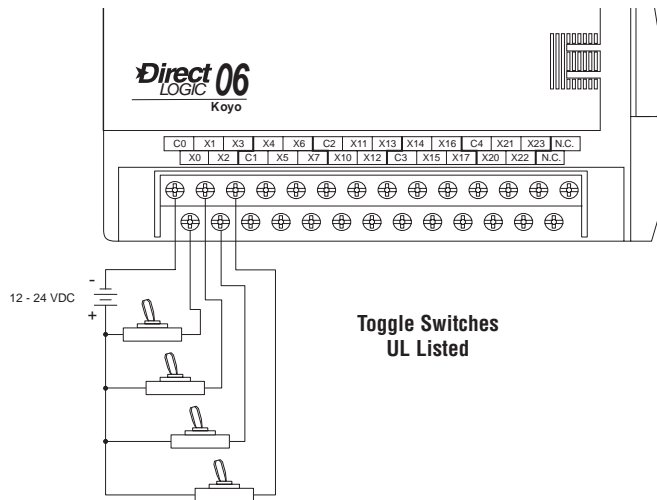
*or*

- D2-HPP Handheld Programmer, firmware version 2.20 or later, (comes with programming cable). Please purchase Handheld Programmer Manual D2-HPP-M separately.

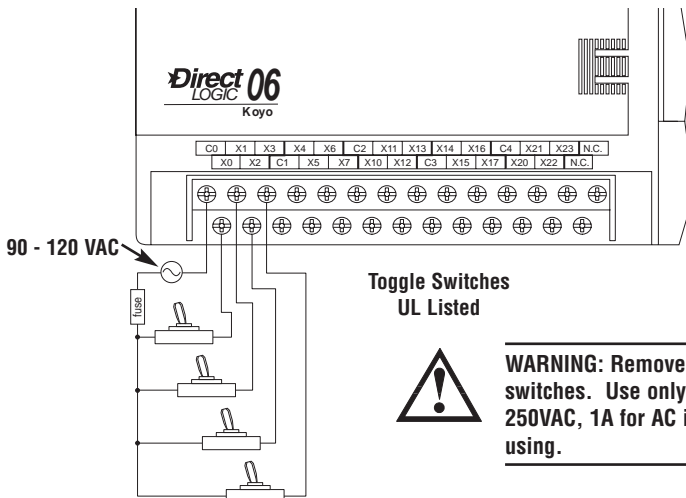


## Step 2: Connect Switches to Input Terminals

To proceed with this quick-start exercise or to follow other examples in this manual, you will need to connect one or more input switches as shown below. If you have DC inputs on an AC-supply DL06, you can use the auxiliary 24VDC supply on the output terminal block or other external 12-24VDC power supply. Be sure to follow the instructions in the accompanying WARNING on this page.



D0-06DA, D0-06DD1,  
D0-06DD2, D0-06DR,  
D0-DD1-D, and D0-06DR1-D  
DC Input



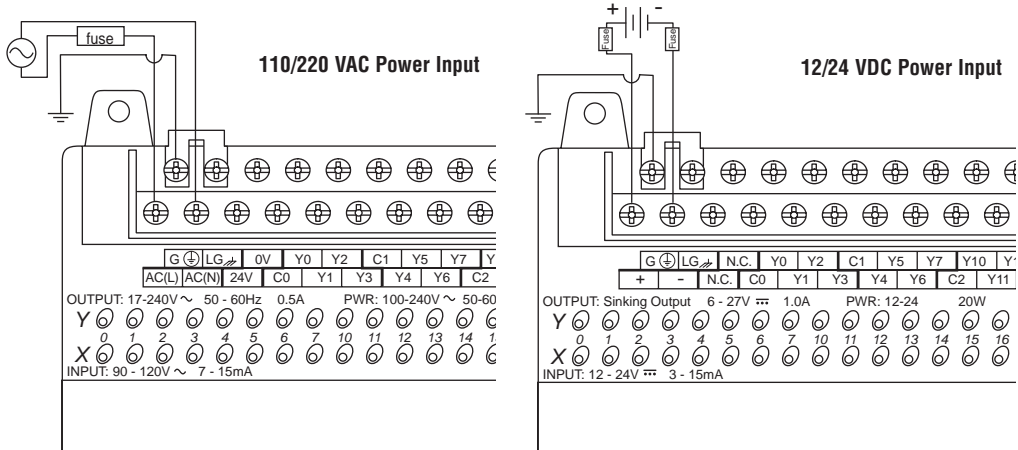
D0-06AA and D0-06AR  
AC input only



**WARNING:** Remove power and unplug the DL06 when wiring the switches. Use only UL-approved switches rated for at least 250VAC, 1A for AC inputs. Firmly mount the switches before using.

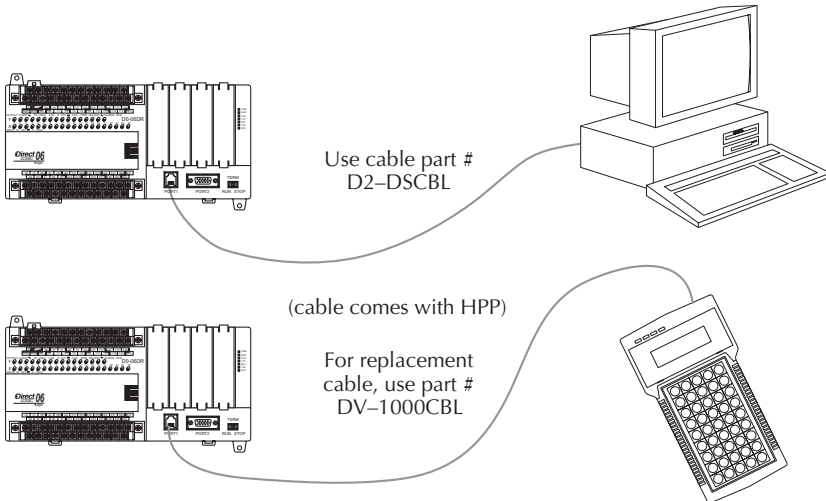
## Step 3: Connect the Power Wiring

Connect the power input wiring for the DL06. Observe all precautions stated earlier in this manual. For more details on wiring, see Chapter 2 on Installation, Wiring, and Specifications. When the wiring is complete, close the connector covers. Do not apply power at this time



## Step 4: Connect the Programming Device

Most programmers will use *DirectSOFT32* programming software, Version 4.0 or later, installed on a personal computer. An alternative, if you need a compact portable programming device, is the Handheld Programmer (firmware version 2.20 or later). Both devices will connect to COM port 1 of the DL06 via the appropriate cable.



*Note: The Handheld Programmer cannot create or access LCD, ASCII or MODBUS instructions.*

Step 5: Switch on the System Power

Apply power to the system and ensure the PWR indicator on the DL06 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 6: Initialize Scratchpad Memory

It's a good precaution to always clear the system memory (scratchpad memory) on a new DL06. There are two ways to clear the system memory:

- In *DirectSOFT32*, select the PLC menu, then Setup, then Initialize Scratchpad. For additional information, see the *DirectSOFT32* Manual. Initializing Scratchpad will return secondary comm port settings and retentive range settings to default. If you have made any changes to these you will need to note these changes and re-enter them after initializing Scratchpad.
- For the Handheld Programmer, use the AUX key and execute AUX 54.

See the Handheld Programmer Manual for additional information.

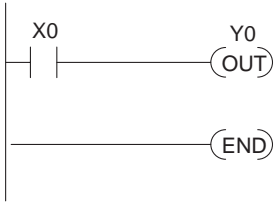
Step 7: Enter a Ladder Program

At this point, *DirectSOFT32* programmers need to refer to the Quick Start Tutorial in the *DirectSOFT32* Manual. There you will learn how to establish a communications link with the DL06 PLC, change CPU modes to Run or Program, and enter a program.

If you are learning how to program with the Handheld Programmer, make sure the CPU is in Program Mode (the RUN LED on the front of the DL06 should be off). If the RUN LED is on, use the MODE key on the Handheld Programmer to put the PLC in Program Mode, then switch to TERM.

Enter the following keystrokes on the Handheld Programmer.

Equivalent *DirectSOFT32* display



CLR

CLR

Clear the Program

C 2

E 4

AUX

ENT

ENT

CLR

Move to the first address and enter X0 contact

NEXT

\$ STR

→

A 0

ENT

Enter output Y0

GX OUT

→

A 0

ENT

Enter the END statement

SHFT

E 4

N TMR

D 3

ENT

After entering the simple example program put the PLC in Run mode by using the Mode key on the Handheld Programmer.

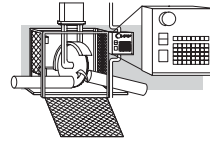
The RUN indicator on the PLC will illuminate indicating the CPU has entered the Run mode. If not, repeat this step, ensuring the program is entered properly or refer to the troubleshooting guide in chapter 9.

After the CPU enters the run mode, the output status indicator for Y0 should follow the switch status on input channel X0. When the switch is on, the output will be on.

# Steps to Designing a Successful System

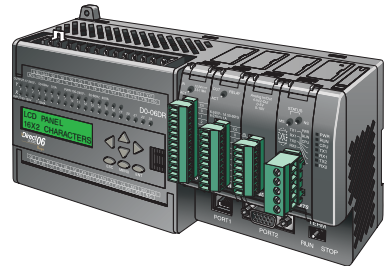
### Step 1: Review the Installation Guidelines

Always make safety the first priority in any system design. Chapter 2 provides several guidelines that will help you design a safer, more reliable system. This chapter also includes wiring guidelines for the various versions of the DL06 PLC.



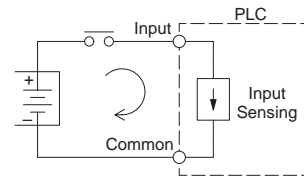
### Step 2: Understand the PLC Setup Procedures

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



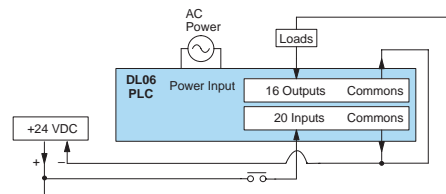
### Step 3: Review the I/O Selection Criteria

There are many considerations involved when you select your I/O type and field devices. Take time to understand how the various types of sensors and loads can affect your choice of I/O type.



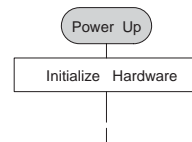
### Step 4: Choose a System Wiring Strategy

It is important to understand the various system design options that are available before wiring field devices and field-side power supplies to the Micro PLC.



### Step 5: Understand the System Operation

Before you begin to enter a program, it is very helpful to understand how the DL06 system processes information. This involves not only program execution steps, but also involves the various modes of operation and memory layout characteristics.

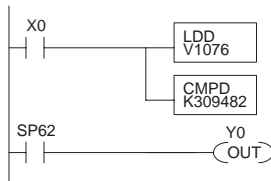


## Step 6: Review the Programming Concepts

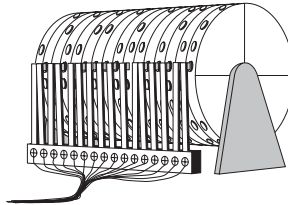
The DL06 PLC instruction set provides for three main approaches to solving the application program, depicted in the figure below.

- 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 also be needed to augment drums and stages.
- The Timer/Event Drum Sequencer features up to 16 steps and offers both time and/or event-based step transitions. The DRUM instruction is best for a repetitive process based on a single series of steps.
- Stage programming (also called RLLPlus) 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.

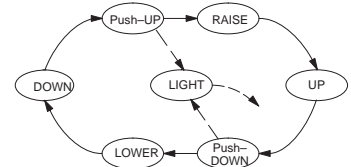
**Standard RLL Programming**  
(see Chapter 5)



**Timer/Event Drum Sequencer**  
(see Chapter 6)



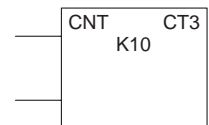
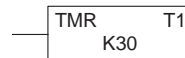
**Stage Programming**  
(see Chapter 7)



After reviewing the programming concepts above, you'll be equipped with a variety of tools to write your application program.

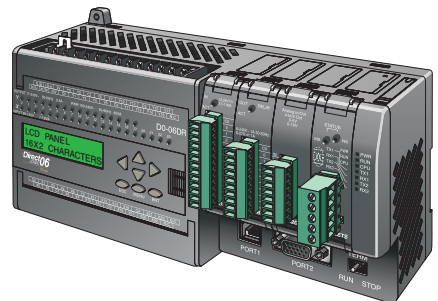
## Step 7: Choose the Instructions

Once you have installed the Micro PLC and understand the main programming concepts, you can begin writing your application program. At that time you will begin to use one of the most powerful instruction sets available in a small PLC.



## Step 8: Understand the Maintenance and Troubleshooting Procedures

Sometimes equipment failures occur when we least expect it. Switches fail, loads short and need to be replaced, etc. In most cases, the majority of the troubleshooting and maintenance time is spent trying to locate the problem. The DL06 Micro PLC has many built-in features such as error codes that can help you quickly identify problems.





# Questions and Answers about DL06 Micro PLCs

### Q. What is the instruction set like?

A. The instruction set is very close to that of our DL260 CPU. The DL06 instructions include the drum sequencing instruction, networking, ASCII, MODBUS, LCD and High-Speed I/O capabilities. High-Speed inputs are available on units with DC inputs only; high-speed outputs are available on units with DC outputs only.

### Q. Do I have to buy the full *DirectSOFT32* programming package to program the DL06?

A. No. We offer a version of *DirectSOFT32* just for our micro PLC products, PC-PGM-BRICK, and it's very affordable.

### Q. Is the DL06 expandable?

A. Yes, the DL06 series function as stand-alone PLCs. However, option card slots allow you to expand the system without changing the footprint.

### Q. Does the DL06 have motion control capability?

A. Yes, the DL06 has limited motion control capabilities. The High-Speed I/O features offer either encoder inputs with high-speed counting and presets with interrupt, or a pulse/direction output for stepper control. Three types of motion profiles are available, which are explained in Chapter 3.

### Q. Are the ladder programs stored in a removable EEPROM?

A. No. The DL06 contains a non-removable FLASH memory for program storage, which may be written and erased thousands of times. You may transfer programs to/from *DirectSOFT32* on a PC.

### Q. Does the DL06 contain fuses for its outputs?

A. There are no output circuit fuses. Therefore, we recommend fusing each channel, or fusing each common. See Chapter 2 for I/O wiring guidelines.

### Q. Is the DL06 Micro PLC U.L. approved?

A. The Micro PLC has met the requirements of UL (Underwriters' Laboratories, Inc.), and CUL (Canadian Underwriters' Laboratories, Inc.).

### Q. Does the DL06 Micro PLC comply with European Union (EU) Directives?

A. The Micro PLC has met the requirements of the European Union Directives (CE).

**Q. Which devices can I connect to the communication ports of the DL06?**

A. Port 1: The port is RS-232C, fixed at 9600 baud, odd parity, address 1, and uses the proprietary K-sequence protocol. The DL06 can also connect to MODBUS RTU and DirectNET networks as a slave device through port 1. The port communicates with the following devices:

- DV-1000 Data Access Unit, EZTouch, EZText, DirectTouch, LookoutDirect, DSData or Optimization Operator interface panels
- *DirectSOFT32* (running on a personal computer)
- D2-HPP handheld programmer
- Other devices which communicate via K-sequence, Directnet, MODBUS RTU protocols should work with the DL06 Micro PLC. Contact the vendor for details.

A. Port 2: This is a multi-function port. It supports RS-232C, RS422, or RS485, with selective baud rates (300-38,400bps), address and parity. It also supports the proprietary K-sequence protocol as well as DirectNet and MODBUS RTU, ASCII In/Out and non-sequence/print protocols.

**Q. Can the DL06 accept 5VDC inputs?**

A. No, 5 volts is lower than the DC input ON threshold. However, many TTL logic circuits can drive the inputs if they are wired as open collector (sinking) inputs. See Chapter 2 for I/O wiring guidelines.

# INSTALLATION, WIRING, AND SPECIFICATIONS

---



## In This Chapter...

Safety Guidelines .....	2-2
Orientation to DL06 Front Panel .....	2-4
Mounting Guidelines .....	2-6
Wiring Guidelines .....	2-10
System Wiring Strategies .....	2-13
Glossary of Specification Terms .....	2-25
Wiring Diagrams and Specifications .....	2-26

# 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.*

---



**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. You should use external electro-mechanical devices, such as relays or limit switches, that are independent of the PLC application to provide protection for any part of the system that may cause personal injury or damage. 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.

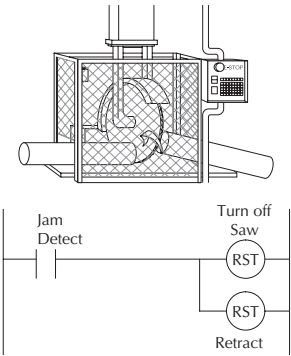
- Orderly system shutdown sequence in the PLC control program
- Mechanical disconnect for output module power
- Emergency stop switch for disconnecting system power

### Orderly System Shutdown

The first level of fault detection is ideally the PLC control program, which can identify machine problems. You must shutdown sequences that must be performed. These 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.



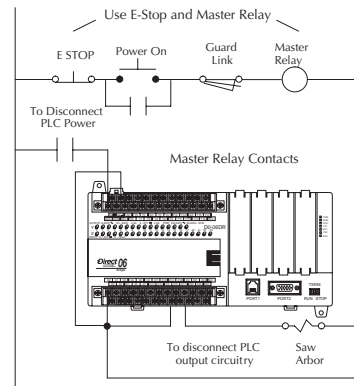
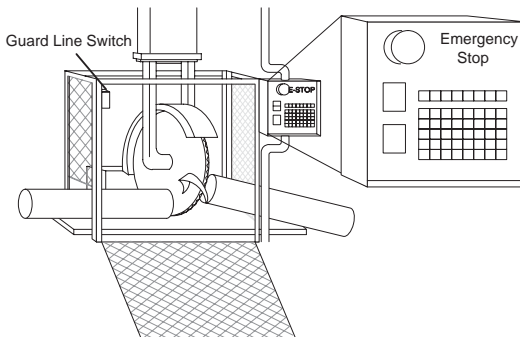
### System Power Disconnect

You should also use electro-mechanical devices, such as master control relays and/or limit switches, to prevent accidental equipment startup at an unexpected time. These devices should be installed in such a manner to prevent any machine operations from occurring.

For example, if the machine has a jammed part the PLC control program can turn off the saw blade and retract the arbor. However, since the operator must open the guard to remove the part, you should also include a bypass switch that disconnects all system power any time the guard is opened.

### Emergency Stop

The machinery must provide a quick manual method of disconnecting all system power. The disconnect device or switch must be clearly labeled “Emergency Stop”.



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.

### Class 1, Division 2 Approval

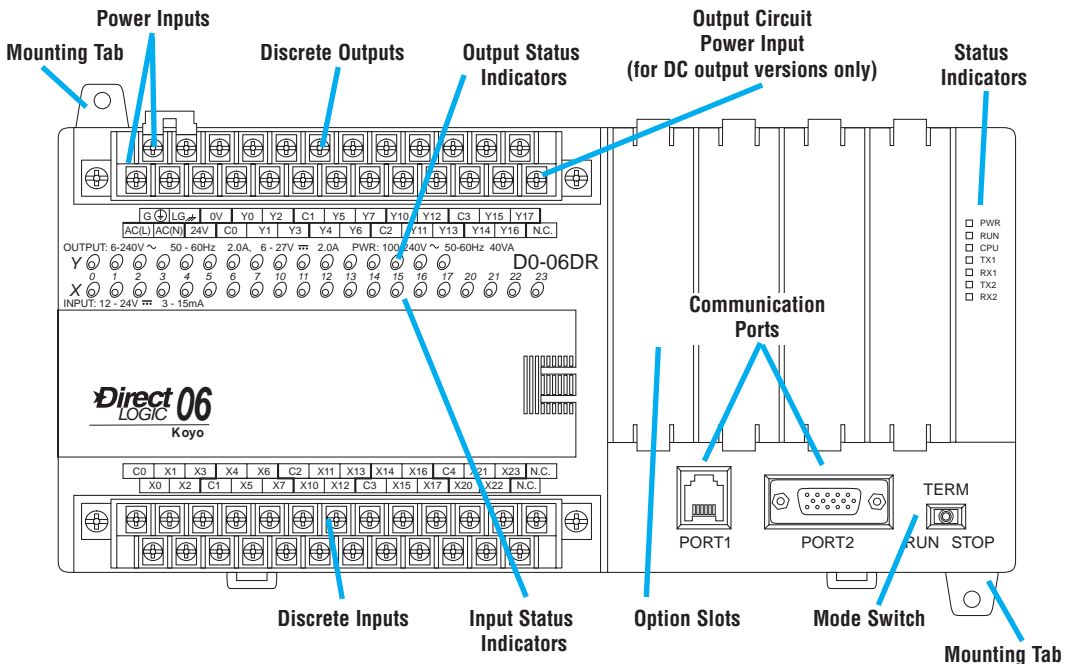
This equipment is suitable for use in Class 1, Division 2, groups A, B, C and D or non-hazardous locations only.



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

### Orientation to DL06 Front Panel

Most connections, indicators, and labels on the DL06 Micro PLCs are located on its front panel. The communication ports are located on front of the PLC as are the option card slots and the mode selector switch. Please refer to the drawing below.



The output and power connector accepts external power and logic and chassis ground connections on the indicated terminals. The remaining terminals are for connecting commons and output connections Y0 through Y17. The sixteen output terminals are numbered in octal, Y0-Y7 and Y10-Y17. On DC output units, the end terminal on the right accepts power for the output stage. The input side connector provides the location for connecting the inputs X0 and X23 and the associated commons.

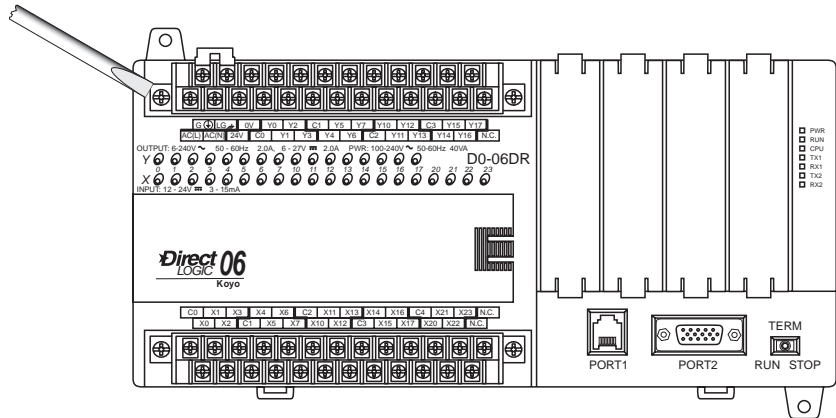


**WARNING: For some applications, field device power may still be present on the terminal block even though the Micro PLC is turned off. To minimize the risk of electrical shock, check all field device power before you expose or remove either connector**

### Terminal Block Removal

The DL06 terminals are divided into two groups. Each group has its own terminal block. The outputs and power wiring are on one block, and the input wiring is on the other. In some instances, it may be desirable to remove the terminal block for easy wiring. The terminal block is designed for easy removal with just a small screwdriver. The drawing below shows the procedure for removing one of the terminal blocks.

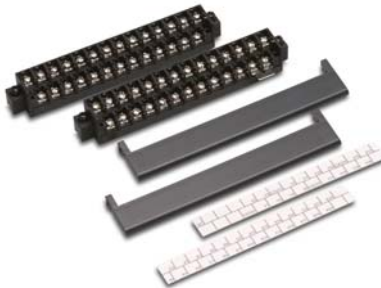
1. Loosen the retention screws on each end of the connector block.



2. From the center of the connector block, pry upward with the screwdriver until the connector is loose.

The terminal blocks on DL06 PLCs have regular (m3 size) screw terminals, which will accept either standard blade-type or #1 Philips screwdriver tips. You can insert one 16 AWG wire under a terminal, or two 18 AWG wires (one on each side of the screw). Be careful not to over-tighten; maximum torque is 6 inch/ounces.

Spare terminal blocks are available in an accessory kit. Please refer to part number D0-ACC-2. You can find this and other accessories on our web site.



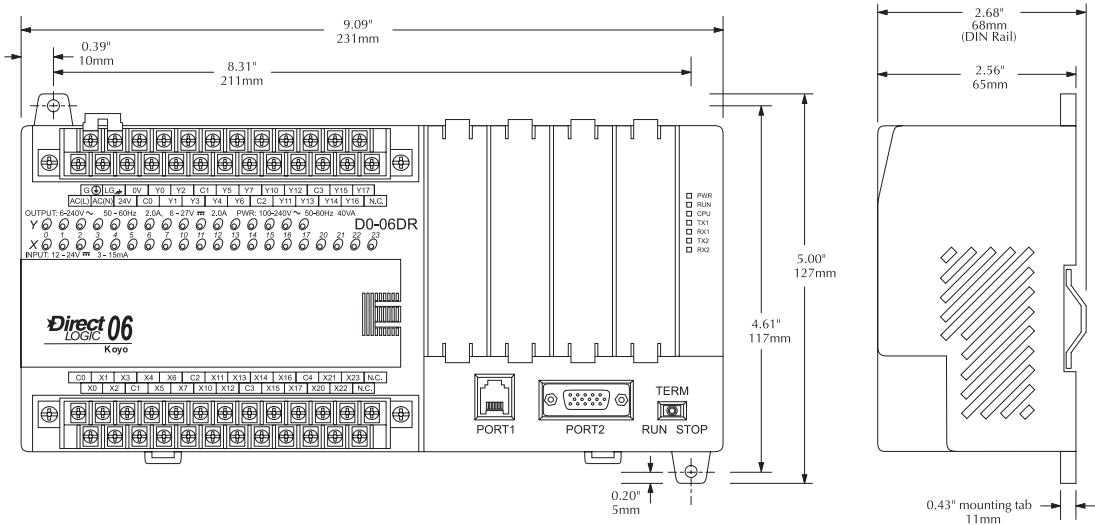
### Mounting Guidelines

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

- Environmental Specifications
- Power Requirements
- Agency Approvals
- Enclosure Selection and Component Dimensions

### Unit Dimensions

The following diagram shows the outside dimensions and mounting hole locations for all versions of the DL06. Make sure you follow the installation guidelines to allow proper spacing from other components.



### Enclosures

Your selection of a proper enclosure is important to ensure safe and proper operation of your DL06 system. Applications of DL06 systems vary and may require additional features. The minimum considerations for enclosures include:

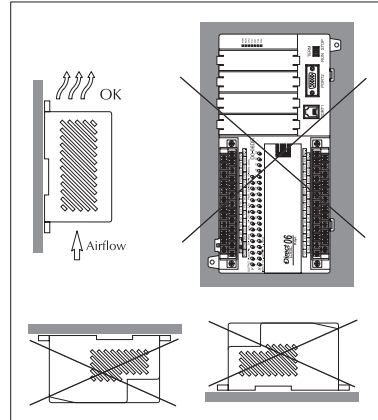
- Conformance to electrical standards
- Protection from the elements in an industrial environment
- Common ground reference
- Maintenance of specified ambient temperature
- Access to equipment
- Security or restricted access
- Sufficient space for proper installation and maintenance of equipment



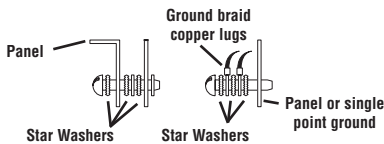
### Panel Layout & Clearances

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. Mount the PLCs horizontally as shown below to provide proper ventilation. You **cannot** mount the DL06 units vertically, upside down, or on a flat horizontal surface. If you place more than one unit in a cabinet, there must be a minimum of 7.2" (183mm) between the units.
2. Provide a minimum clearance of 1.5" (39mm) between the unit and all sides of the cabinet. Note, remember to allow for any operator panels or other items mounted in the door.
3. There should also be at least 3" (78mm) of clearance between the unit and any wiring ducts that run parallel to the terminals.

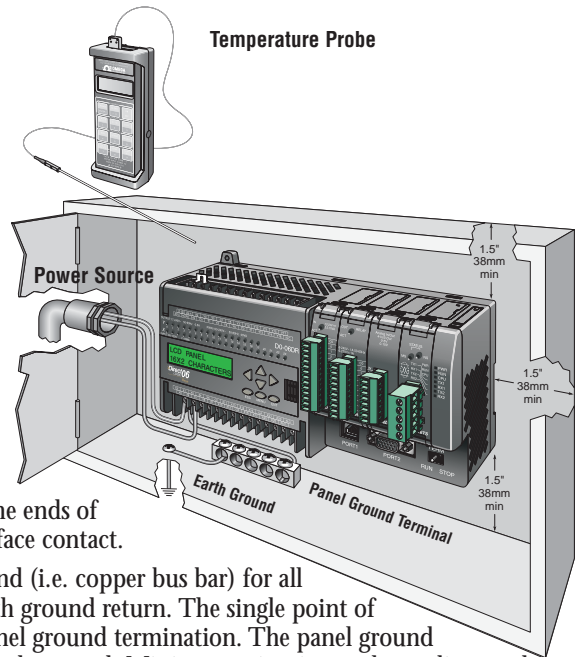


*Note: There is a minimum clearance requirement of 1.5" (38mm) between the panel door (or any devices mounted in the panel door) and the nearest DL06 component.*



4. The ground terminal on the DL06 base must be connected to a single point ground. Use copper stranded wire to achieve a low impedance. Copper eye lugs should be crimped and soldered to the ends of the stranded wire to ensure good surface contact.

5. 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. Minimum wire sizes, color coding, and general safety practices should comply with appropriate electrical codes and standards for your area.



6. A good common ground reference (Earth ground) is essential for proper operation of the DL06. One side of all control and power circuits and the ground lead on flexible shielded cable must be properly connected to Earth ground. There are several methods of providing an adequate common ground reference, including:

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

7. Evaluate any installations where the ambient temperature may approach the lower or upper limits of the specifications. If you suspect the ambient temperature will not be within the operating specification for the DL06 system, measures such as installing a cooling/heating source must be taken to get the ambient temperature within the range of specifications.

8. The DL06 systems are designed to be powered by 95-240 VAC or 12-24 VDC normally available throughout an industrial environment. Electrical power in some areas where the PLCs are installed is not always stable and storms can cause power surges. Due to this, powerline filters are recommended for protecting the DL06 PLCs from power surges and EMI/RFI noise. The Automation Powerline Filter, for use with 120 VAC and 240 VAC, 1-5 Amps, is an excellent choice, however, you can use a filter of your choice. These units install easily between the power source and the PLC.



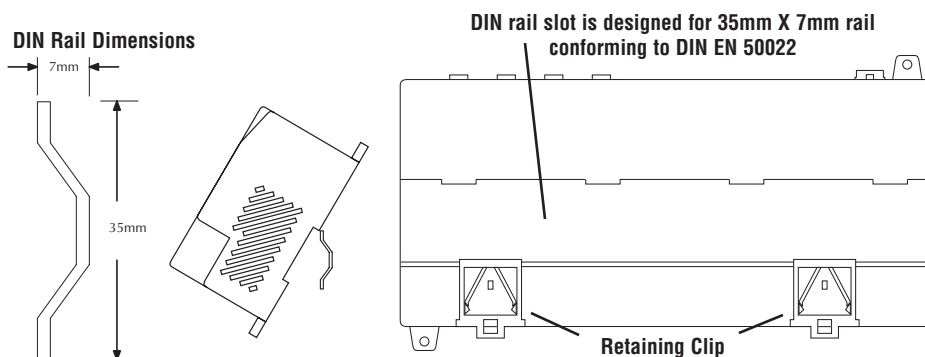
---

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

---

### Using Mounting Rails

DL06 Micro PLCs can be secured to a panel by using mounting rails. We recommend rails that conform to DIN EN standard 50 022. They are approximately 35mm high, with a depth of 7mm. If you mount the Micro PLC on a rail, do consider using end brackets on each side of the PLC. The end bracket helps keep the PLC from sliding horizontally along the rail, reducing the possibility of accidentally pulling the wiring loose. On the bottom of the PLC are two small retaining clips. To secure the PLC to a DIN rail, place it onto the rail and gently push up on the clips to lock it onto the rail. To remove the PLC, pull down on the retaining clips, lift up on the PLC slightly, then pull it away from the rail.



**NOTE:** Refer to our catalog or web site for a complete listing of **DINector** connection systems.

### Environmental Specifications

The following table lists the environmental specifications that generally apply to DL06 Micro PLCs. The ranges that vary for the Handheld Programmer are noted at the bottom of this chart. Certain output circuit types may have derating curves, depending on the ambient temperature and the number of outputs ON. Please refer to the appropriate section in this chapter pertaining to your particular DL06 PLC.

Environmental Specifications	
Specification	Rating
Storage temperature	−4° F to 158° F (−20° C to 70° C)
Ambient operating temperature*	32° F to 131° F (0° C to 55° C)
Ambient humidity**	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
Agency approvals	UL, CE (C1D2), FCC class A

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

\*\*Equipment will operate down to 5% relative humidity. However, static electricity problems occur much more frequently at low humidity levels (below 30%). Make sure you take adequate precautions when you touch the equipment. Consider using ground straps, anti-static floor coverings, etc. if you use the equipment in low-humidity environments.

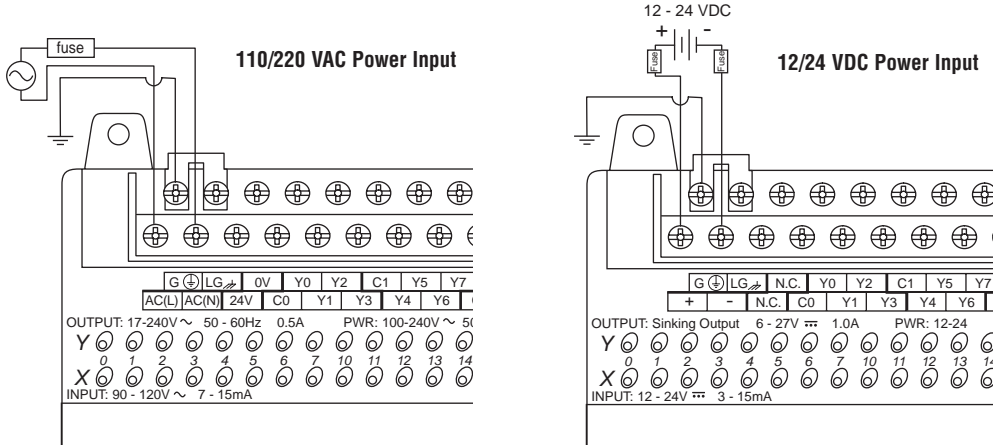
### Agency Approvals

Some applications require agency approvals for particular components. The DL06 Micro PLC agency approvals are listed below:

- UL (Underwriters' Laboratories, Inc.)
- CUL (Canadian Underwriters' Laboratories, Inc.)
- CE (European Economic Union)

Wiring Guidelines

Connect the power input wiring for the DL06. Observe all precautions stated earlier in this manual. For more details on wiring, see Chapter 2 on Installation, Wiring, and Specifications. When the wiring is complete, close the connector covers. Do not apply power at this time.



**WARNING:** Once the power wiring is connected, secure the terminal block cover in the closed position. When the cover is open there is a risk of electrical shock if you accidentally touch the connection terminals or power wiring.

Fuse Protection for Input Power

There are no internal fuses for the input power circuits, so external circuit protection is needed to ensure the safety of service personnel and the safe operation of the equipment itself. To meet UL/CUL specifications, the input power must be fused. Depending on the type of input power being used, follow these fuse protection recommendations:

208/240 VAC Operation

When operating the unit from 208/240 VAC, whether the voltage source is a step-down transformer or from two phases, fuse both the line (L) and neutral (N) leads. The recommended fuse size is 1.0A (fast blow).

110/125 VAC Operation

When operating the unit from 110/125 VAC, it is only necessary to fuse the line (L) lead; it is not necessary to fuse the neutral (N) lead. The recommended fuse size is 1.0A (fast blow).

### 12/24 VDC Operation

When operating at these lower DC voltages, wire gauge size is just as important as proper fusing techniques. Using large conductors minimizes the voltage drop in the conductor. Each DL06 input power terminal can accommodate one 16 AWG wire or two 18 AWG wires. A DC failure can maintain an arc for much longer time and distance than AC failures. Typically, the main bus is fused at a higher level than the branch device, which in this case is the DL06. The recommended fuse size for the branch circuit to the DL06 is 1.5A (for example, a Littlefuse 312.001 or equivalent).

### External Power Source

The power source must be capable of supplying voltage and current complying with individual Micro PLC specifications, according to the following specifications:

Power Source Specifications		
Item	DL06 VAC Powered Units	DL06 VDC Powered Units
Input Voltage Range	110/220 VAC (95–240 VAC)	12–24 VDC (10.8–26.4 VDC)
Maximum Inrush Current	13 A, 1ms (95–240 VAC) 15 A, 1ms (240–264 VAC)	10A
Maximum Power	30 VA	20 W
Voltage Withstand (dielectric)	1 minute @ 1500 VAC between primary, secondary, field ground	
Insulation Resistance	> 10 MΩ at 500 VDC	

**NOTE:** The rating between all internal circuits is BASIC INSULATION ONLY.



### Planning the Wiring Routes

The following guidelines provide general information on how to wire the I/O connections to DL06 Micro PLCs. For specific information on wiring a particular PLC refer to the corresponding specification sheet which appears later in this chapter.

- Each terminal connection of the DL06 PLC can accept one 16 AWG wire or two 18 AWG size wires. Do not exceed this recommended capacity.

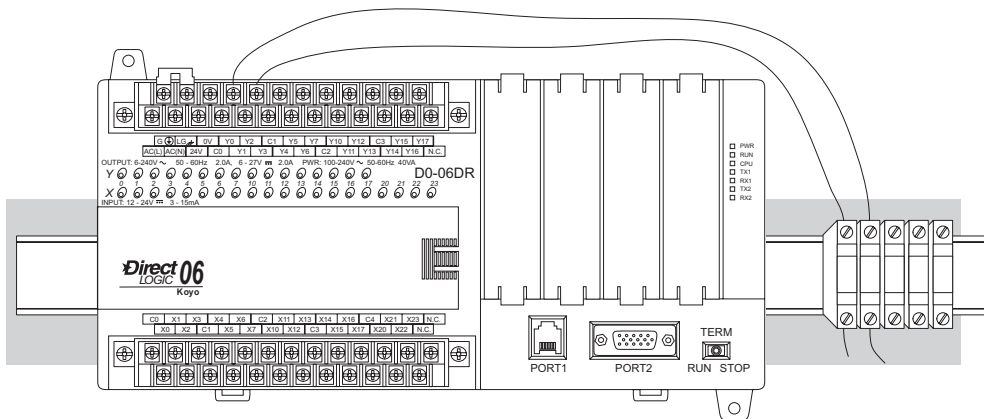


**NOTE:** Recommended wire size for field devices is 16 - 22 AWG solid/stranded. Tighten terminal screws to 7.81 lb-in (0.882 N\*m) to 9.03 lb-in (1.02 N\*m).

- Always use a continuous length of wire. Do not splice wires to attain a needed length.
- Use the shortest possible wire length.
- Use wire trays for routing where possible.
- Avoid running wires near high energy wiring.
- Avoid running input wiring close to output wiring where possible.
- To minimize voltage drops when wires must run a long distance, consider using multiple wires for the return line.
- Avoid running DC wiring in close proximity to AC wiring where possible.
- Avoid creating sharp bends in the wires.
- Install the recommended powerline filter to reduce power surges and EMI/RFI noise.

## Fuse Protection for Input and Output Circuits

Input and Output circuits on DL06 Micro PLCs do not have internal fuses. In order to protect your Micro PLC, we suggest you add external fuses to your I/O wiring. A fast-blow fuse, with a lower current rating than the I/O bank's common current rating can be wired to each common. Or, a fuse with a rating of slightly less than the maximum current per output point can be added to each output. Refer to the Micro PLC specification sheets further in this chapter to find the maximum current per output point or per output common. Adding the external fuse does not guarantee the prevention of Micro PLC damage, but it will provide added protection.



## I/O Point Numbering

All DL06 Micro PLCs have a fixed I/O configuration. It follows the same octal numbering system used on other *DirectLogic* family PLCs, starting at X0 and Y0. The letter X is always used to indicate inputs and the letter Y is always used for outputs.

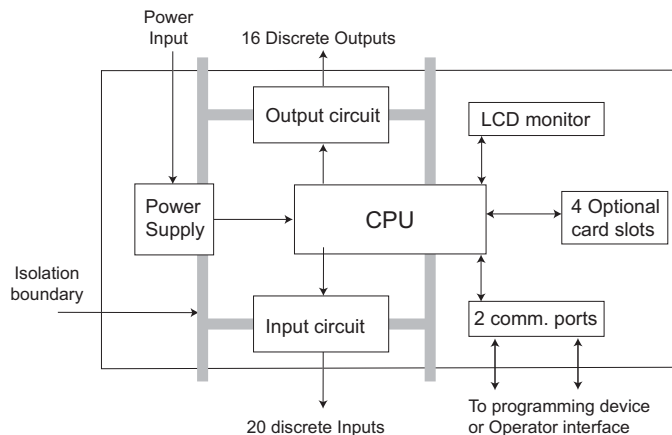
The I/O numbering always starts at zero and does not include the digits 8 or 9. The addresses are typically assigned in groups of 8 or 16, depending on the number of points in an I/O group. For the DL06 the twenty inputs use reference numbers X0 – X23. The sixteen output points use references Y0 – Y17.

## System Wiring Strategies

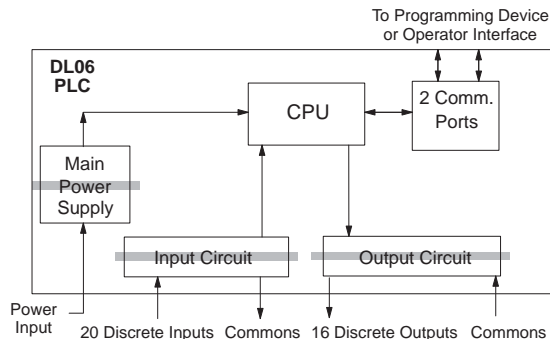
The DL06 Micro PLC is very flexible and will work in many different wiring configurations. By studying this section before actual installation, you can probably find the best wiring strategy for your application. This will help to lower system cost, wiring errors, and avoid safety problems.

### PLC Isolation Boundaries

PLC circuitry is divided into three main regions separated by isolation boundaries, shown in the drawing below. Electrical isolation provides safety, so that a fault in one area does not damage another. A powerline filter will provide isolation between the power source and the power supply. A transformer in the power supply provides magnetic isolation between the primary and secondary sides. Opto-couplers provide optical isolation in Input and Output circuits. This isolates logic circuitry from the field side, where factory machinery connects. Note 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.*



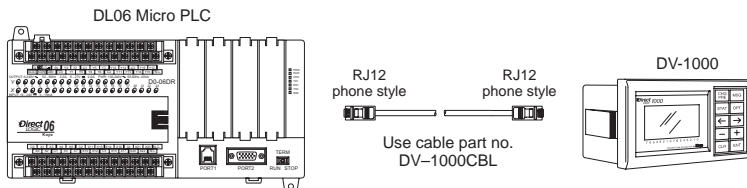
The next figure shows the internal layout of DL06 PLCs, as viewed from the front panel.



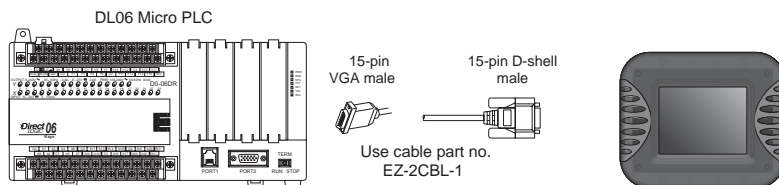
### Connecting Operator Interface Devices

Operator interfaces require data and power connections. Operator interfaces with a large CRT usually require separate AC power. However, small operator interface devices like the popular DV-1000 Data Access Unit may be powered directly from the DL06 Micro PLC.

Connect the DV-1000 to communication port 1 on the DL06 Micro PLC using the cable shown below. A single cable contains transmit/receive data wires and +5V power.

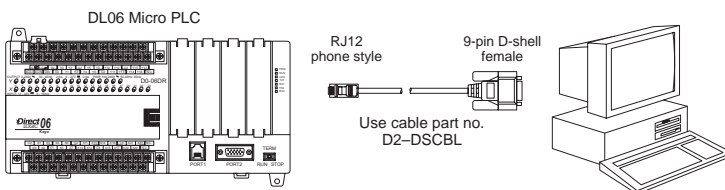


EZ-Touch and EZ-Text operator interface panels require separate power and communications connections. Connect the DL06 to the proper D-shell connector on the rear of the operator panel using the cable shown below. These panels require 8–30VDC power.

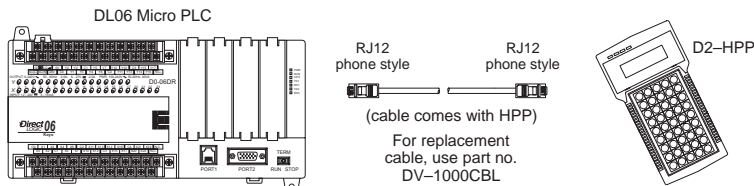


### Connecting Programming Devices

DL06 Micro PLCs can be programmed with either a handheld programmer or with *DirectSOFT32* on a PC. Connect the DL06 to a PC using the cable shown below.



The D2-HPP Handheld Programmer comes with a communications cable. For a replacement part, use the cable shown below.





### Sinking / Sourcing Concepts

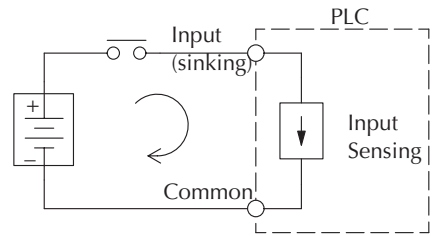
Before going further in our presentation of wiring strategies, we need to introduce the concepts of “sinking” and “sourcing.” These terms apply to typical input or output circuits. It is the goal of this section to make these concepts easy to understand. First we give the following short definitions, followed by practical applications.

**Sinking = Path to supply ground (–)**

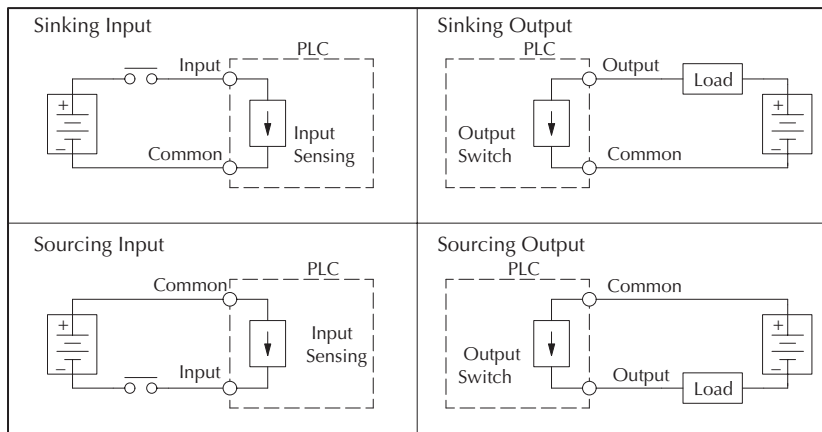
**Sourcing = Path to supply source (+)**

Notice the reference to (+) and (–) polarities. *Sinking and sourcing terminology applies only to DC input and output circuits.* Input and output points that are either sinking or sourcing 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 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 DC-powered DL06 Micro PLCs have selectable sinking or sourcing inputs and either sinking or sourcing outputs. Any pair of input/output circuits shown below is possible with one of the DL06 models.

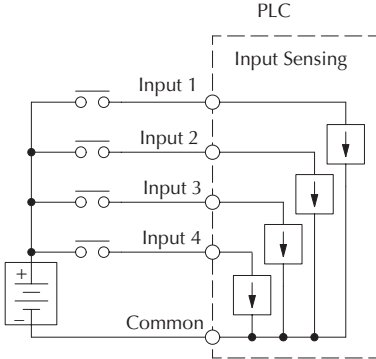
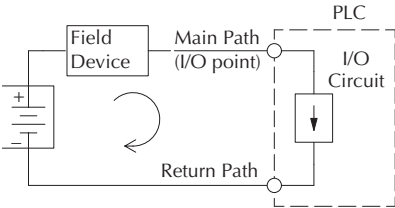


# I/O “Common” Terminal Concepts

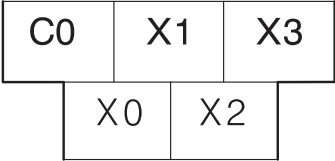
In order for a PLC I/O circuit to operate, current must enter at one terminal and exit at another. This means at least two terminals are associated with every I/O point. In the figure to the right, the input or output terminal is the *main path* for the current. One additional terminal must provide the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

Most input or output point groups on PLCs share the return path among two or more I/O points. 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 to the right, 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 DL06 input and output circuits are grouped into banks that share a common return path. The best indication of I/O common grouping is on the wiring label. The I/O common groups are separated by a bold line. A thinner line separates the inputs associated with that common. To the right, notice that X0, X1, X2, and X3 share the common terminal C0, located to the left of X1.



The following complete set of labels shows five banks of four inputs and four banks of four outputs. One common is provided for each bank.

G	⊕	LG	0V	Y0	Y2	C1	Y5	Y7	Y10	Y12	C3	Y15	Y17
AC(L)	AC(N)	24V	C0	Y1	Y3	Y4	Y6	C2	Y11	Y13	Y14	Y16	N.C.

C0	X1	X3	X4	X6	C2	X11	X13	X14	X16	C4	X21	X23	N.C.
X0	X2	C1	X5	X7	X10	X12	C3	X15	X17	X20	X22	N.C.	

This set of labels is for DC (sinking) output versions such as the D0-06DD1 and D0-06DD1-D. One common is provided for each group of four outputs, and one designated terminal on the output side accepts power for the output stage.

G	⊕	LG	0V	Y0	Y2	C1	Y5	Y7	Y10	Y12	C3	Y15	Y17
AC(L)	AC(N)	24V	C0	Y1	Y3	Y4	Y6	C2	Y11	Y13	Y14	Y16	+V

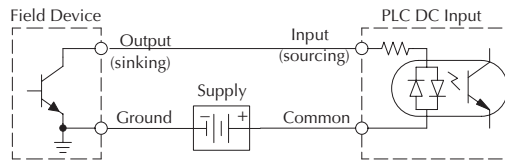
C0	X1	X3	X4	X6	C2	X11	X13	X14	X16	C4	X21	X23	N.C.
X0	X2	C1	X5	X7	X10	X12	C3	X15	X17	X20	X22	N.C.	

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

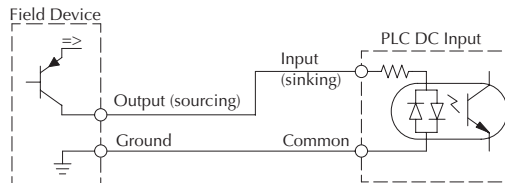
In the previous section on sinking and sourcing concepts, we discussed DC I/O circuits that only allow current to flow one way. This is also true for many of the field devices which have solid-state (transistor) interfaces. In other words, field devices can also be sourcing or sinking. *When connecting two devices in a series DC circuit (as is the case when wiring a field device to a PLC DC input or output), one must be wired as sourcing and the other as sinking.*

#### Solid State Input Sensors

The DL06's DC inputs 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 included auxiliary 24 VDC power supply or another supply (+12 VDC or +24VDC), as long as the input specifications are met.



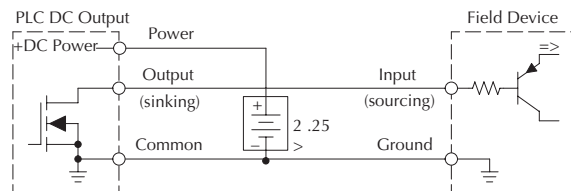
In the next circuit, a field device has an open-emitter PNP transistor output. It sources current to the PLC input point, which sinks the current back to ground. Since the field device is sourcing current, no additional power supply is required between the device and the PLC DC Input.



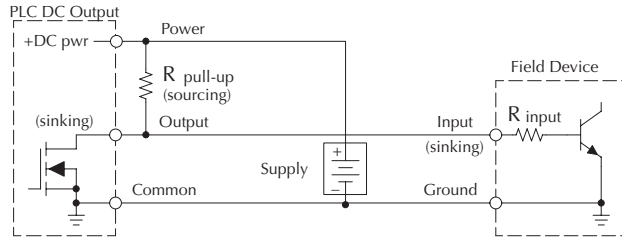
#### Solid State Output Loads

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 signal, not to send DC power to an actuator.

The DL06 PLC family offers DC outputs that are sinking only or DC outputs that are sourcing. All sixteen outputs have the same electrical common, even though there are four common terminal screws. 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 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 Rpull-up from the output to the DC output circuit power input.



**NOTE:** 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-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-up. Then calculate the power Ppull-up (in watts), in order to size R pull-up properly.

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

$$R_{\text{pull-up}} = \frac{V_{\text{supply}} - 0.7}{I_{\text{input}}} - R_{\text{input}}$$

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

## Relay Output Wiring Methods

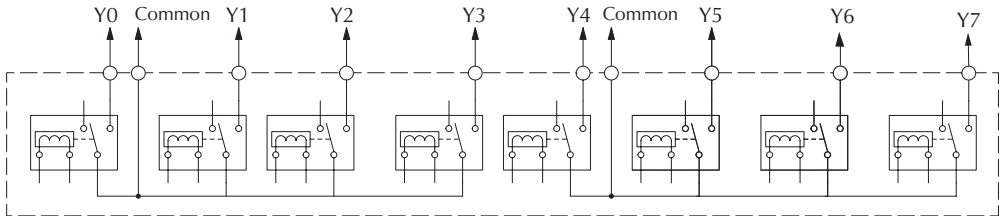
The D0-06AR and the D0-06DR models feature relay outputs. Relays are best for the following applications:

- Loads that require higher currents than the solid-state DL06 outputs can deliver
- Cost-sensitive applications
- Some output channels need isolation from other outputs (such as when some loads require AC while others require DC)

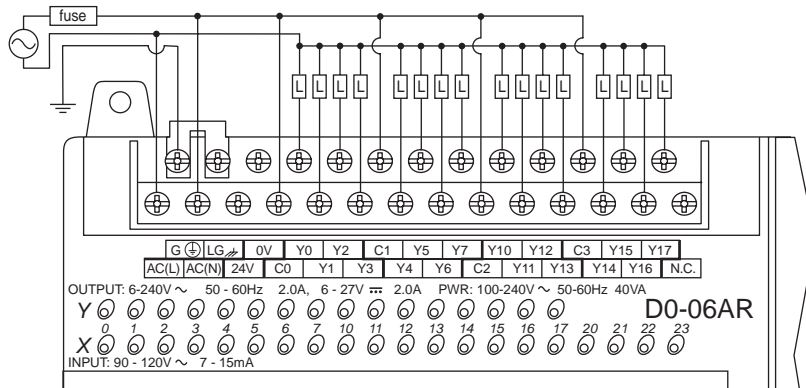
Some applications in which NOT to use relays:

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

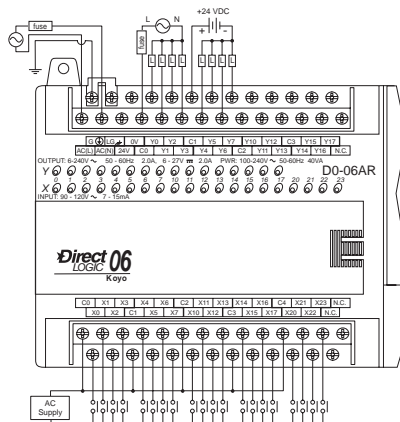
This section presents various ways to wire relay outputs to the loads. The relay output DL06s have sixteen normally-open SPST relays available. They are organized with four relays per common. The figure below shows the relays and the internal wiring of the PLC. Note that each group is isolated from the other group of outputs.



In the circuit below, all loads use the same AC power supply which powers the DL06 PLC. In this example, all commons are connected together.



In the circuit on the following page, loads for Y0 – Y3 use the same AC power supply which powers the DL06 PLC. Loads for Y4 – Y7 use a separate DC supply. In this example, the commons are separated according to which supply powers the associated load.



### Surge Suppression For Inductive Loads

Inductive load devices (devices with a coil) generate transient voltages when de-energized with a relay contact. When a relay contact is closed it “bounces”, which energizes and de-energizes the coil until the “bouncing” stops. The transient voltages generated are much larger in amplitude than the supply voltage, especially with a DC supply voltage.

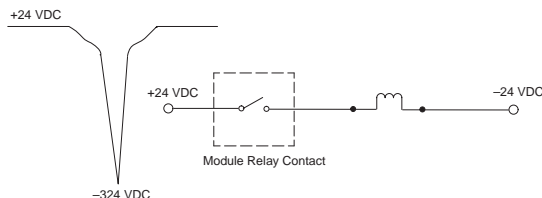
When switching a DC-supplied inductive load the full supply voltage is always present when the relay contact opens (or “bounces”). When switching an AC-supplied inductive load there is one chance in 60 (60 Hz) or 50 (50 Hz) that the relay contact will open (or “bounce”) when the AC sine wave is zero crossing. If the voltage is not zero when the relay contact opens there is energy stored in the inductor that is released when the voltage to the inductor is suddenly removed. This release of energy is the cause of the transient voltages.

When inductive load devices (motors, motor starters, interposing relays, solenoids, valves, etc.) are controlled with relay contacts, it is recommended that a surge suppression device be connected directly across the coil of the field device. If the inductive device has plug-type connectors, the suppression device can be installed on the terminal block of the relay output.

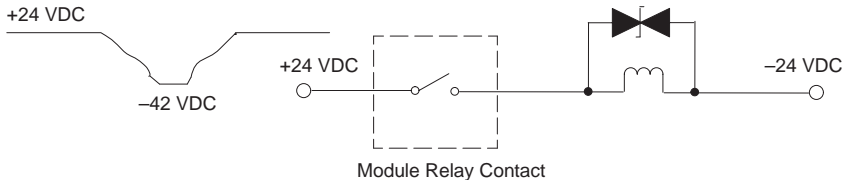
Transient Voltage Suppressors (TVS or transorb) provide the best surge and transient suppression of AC and DC powered coils, providing the fastest response with the smallest overshoot.

Metal Oxide Varistors (MOV) provide the next best surge and transient suppression of AC and DC powered coils.

For example, the waveform in the figure below shows the energy released when opening a contact switching a 24 VDC solenoid. Notice the large voltage spike.



This figure shows the same circuit with a transorb (TVS) across the coil. Notice that the voltage spike is significantly reduced.



Use the following table to help select a TVS or MOV suppressor for your application based on the inductive load voltage.

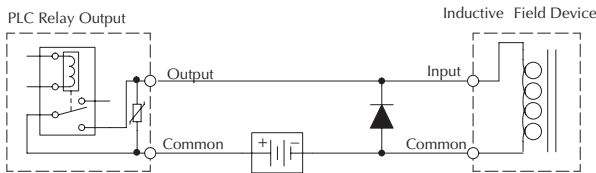
Surge Suppressors			
Vendor / Catalog	Type	Inductive Load Voltage	Part Number
General Instrument Transient Voltage Suppressors, LiteOn Diodes; from DigiKey Catalog; Phone: 1-800-344-4539	TVS	110/120 VAC	ZL-TD8-120
	TVS	24 VDC	ZL-TD8-24
	TVS	110/120 VAC	P6KE180CAGICT-ND
	TVS	220/240 VDC	P6KE350CA
	TVS Diode	12/24 VDC or VAC 12/24 VDC or VAC	P6K30CAGICT-ND 1N4004CT-ND
Harris Metal Oxide Varistors;  from Newark Catalog: Phone 1-800-463-9275	MOV	110/120 VAC	V150LA20C
	MOV	220/240 VAC	V250LA20C

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.

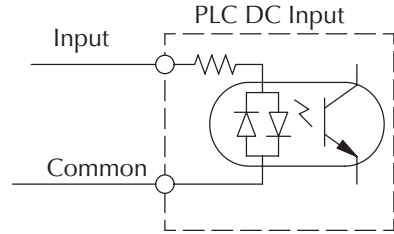
For inductive loads in DC circuits we recommend using a suppression diode as shown in the following diagram (DO NOT use this circuit with an AC power supply). When the load is energized the diode is reverse-biased (high impedance). When the load is turned off, energy stored in its coil is released in 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.

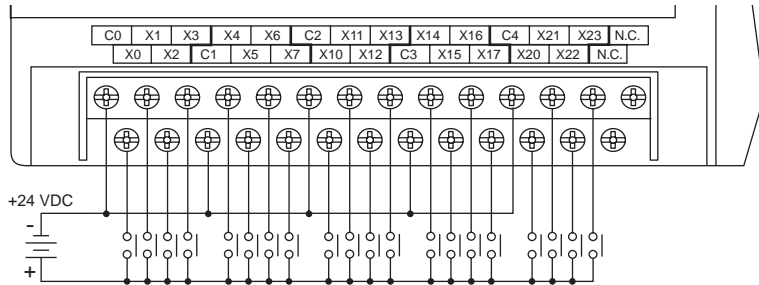


### DC Input Wiring Methods

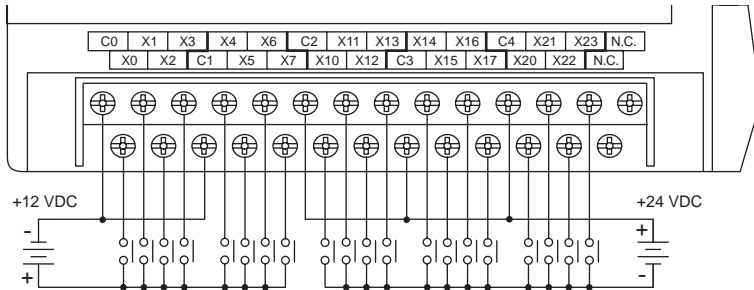
DL06 Micro PLCs with DC inputs are particularly flexible because they can be wired as either sinking or sourcing. The dual diodes (shown to the right) allow 10.8 – 26.4 VDC. The target applications are +12 VDC and +24 VDC. You can actually wire each group of inputs associated common group of inputs as DC sinking and the other half as DC sourcing. Inputs grouped by a common must be all sinking or all sourcing.



In the first and simplest example below, all commons are connected together and all inputs are sinking.



In the next example, the first eight inputs are sinking, and the last twelve are sourcing.

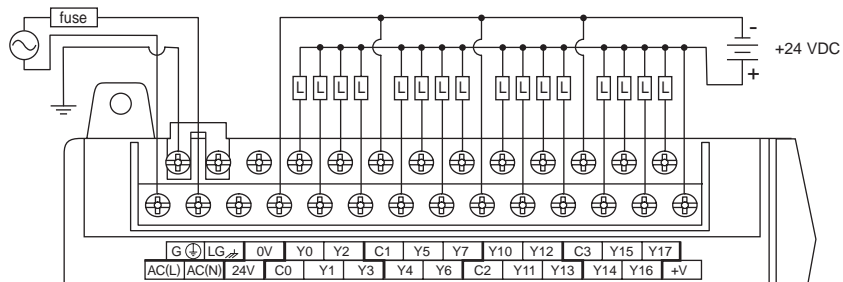




## DC Output Wiring Methods

DL06 DC output circuits are high-performance transistor switches with low on-resistance and fast switching times. Please note the following characteristics which are unique to the DC output type:

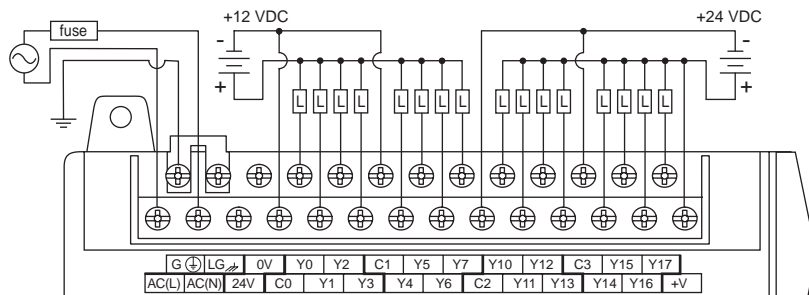
- There is only one electrical common for all sixteen outputs. All sixteen outputs belong to one bank.
- The output switches are current-sinking only or current sourcing only. ***Refer to the detailed specifications in this manual to determine which type output is present on a particular model.***
- The output circuit inside the PLC requires external power. The supply (–) must be connected to a common terminal, and the supply (+) connects the the right-most terminal on the upper connector (+V).



In the example below, all sixteen outputs share a common supply.

In the next example below, the outputs have “split” supplies. The first three outputs are using a +12 VDC supply, and the last three are using a +24 VDC supply. However, you can split the outputs among any number of supplies, as long as:

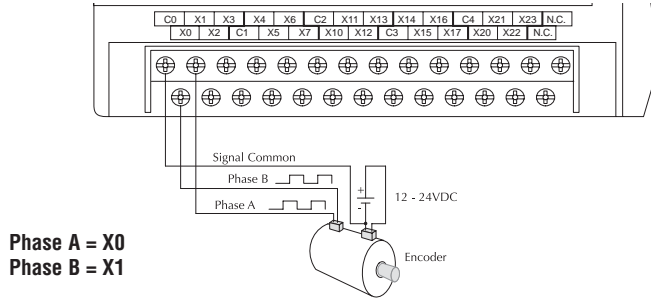
- all supply voltages are within the specified range
- all output points are wired as sinking



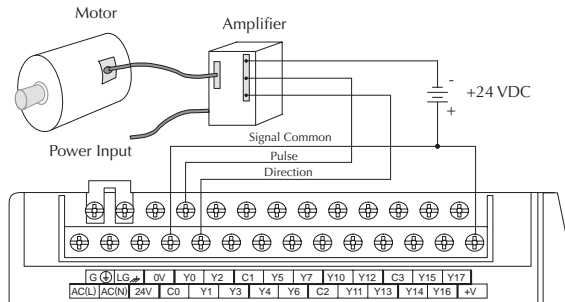
- all source (–) terminals are connected together

### High-Speed I/O Wiring Methods

DL06 versions with DC type input or output points contain a dedicated High-Speed I/O circuit (HSIO). The circuit configuration is programmable, and it processes specific I/O points independently from the CPU scan. Chapter 3 discusses the programming options for HSIO. While the HSIO circuit has six modes, we show wiring diagrams for two of the most popular modes in this chapter. The high-speed input interfaces to points X0 – X3. Properly configured, the DL06 can count quadrature pulses at up to 7 kHz from an incremental encoder as shown below.



DL06 versions with DC type output points can use the High Speed I/O Pulse Output feature. It can generate high-speed pulses at up to 10 kHz for specialized control such as stepper motor / intelligent drive systems. Output Y0 and Y1 can generate pulse and direction signals, or it can generate CCW and CW pulse signals respectively. See Chapter 3 on high-speed input and pulse output options.



## Glossary of Specification Terms

### **Discrete Input**

One of twenty input connections to the PLC which converts an electrical signal from a field device to a binary status (off or on), which is read by the internal CPU each PLC scan.

### **Discrete Output**

One of sixteen output connections from the PLC which converts an internal ladder program result (0 or 1) to turn On or Off an output switching device. This enables the program to turn on and off large field loads.

### **I/O Common**

A connection in the input or output terminals which is shared by multiple I/O circuits. It usually is in the return path to the power supply of the I/O circuit.

### **Input Voltage Range**

The operating voltage range of the input circuit.

### **Maximum Voltage**

Maximum voltage allowed for the input circuit.

### **ON Voltage Level**

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

### **OFF Voltage Level**

The maximum voltage level at which the input point will turn OFF

### **Input Impedance**

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

### **Input Current**

Typical operating current for an active (ON) input.

### **Minimum ON Current**

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

### **Maximum OFF Current**

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

### **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. All LEDs on DL06 Micro PLCs are electrically located on the logic side of the input or output circuit.

# Wiring Diagrams and Specifications

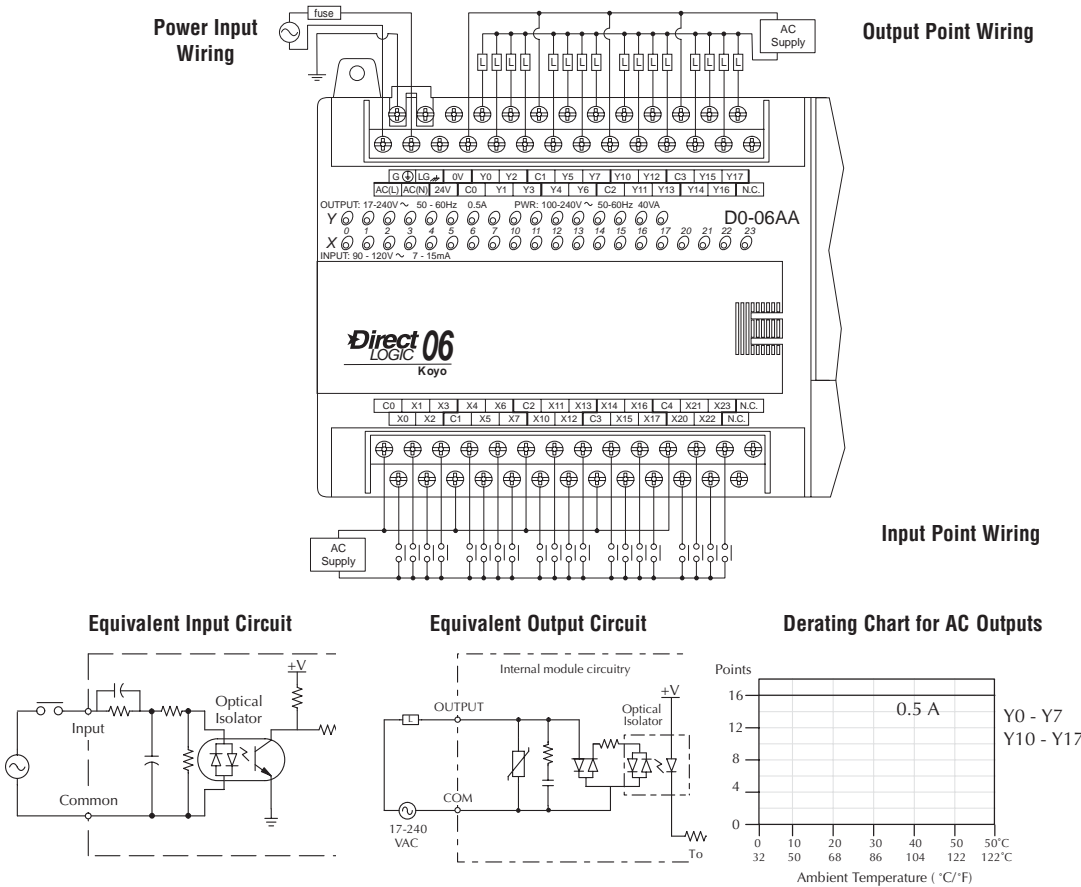
The remainder of this chapter provides detailed technical information for the DL06 PLCs. A basic wiring diagram, equivalent I/O circuits, and specification tables are laid out for each PLC.

## D0-06AA I/O Wiring Diagram

The D0-06AA PLC has twenty AC inputs and sixteen AC outputs. The following diagram shows a typical field wiring example. The AC external power connection uses four terminals as shown.

Inputs are organized into five banks of four. Each bank has an isolated common terminal. The wiring example below shows all commons connected together, but separate supplies and common circuits may be used. The equivalent input circuit shows one channel of a typical bank.

Outputs are organized into four banks of four triac switches. Each bank has a common terminal. The wiring example below shows all commons connected together, but separate supplies and common circuits may be used. The equivalent output circuit shows one channel of a typical bank.



D0-06AA General Specifications	
<b>External Power Requirements</b>	100– 240 VAC, 40 VA maximum,
<b>Communication Port 1 9600 baud (Fixed), 8 data bits, 1 stop bit odd parity</b>	K–Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
<b>Communication Port 2 9600 baud (default) 8 data bits, 1 stop bit odd parity</b>	K–Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / print, ASCII in/out
<b>Programming cable type</b>	D2–DSCBL
<b>Operating Temperature</b>	32 to 131° F (0 to 55 C)
<b>Storage Temperature</b>	–4 to 158° F (–20 to 70 C)
<b>Relative Humidity</b>	5 to 95% (non-condensing)
<b>Environmental air</b>	No corrosive gases permitted
<b>Vibration</b>	MIL STD 810C 514.2
<b>Shock</b>	MIL STD 810C 516.2
<b>Noise Immunity</b>	NEMA ICS3–304
<b>Terminal Type</b>	Removable
<b>Wire Gauge</b>	One AWG16 or two AWG18, AWG24 minimum

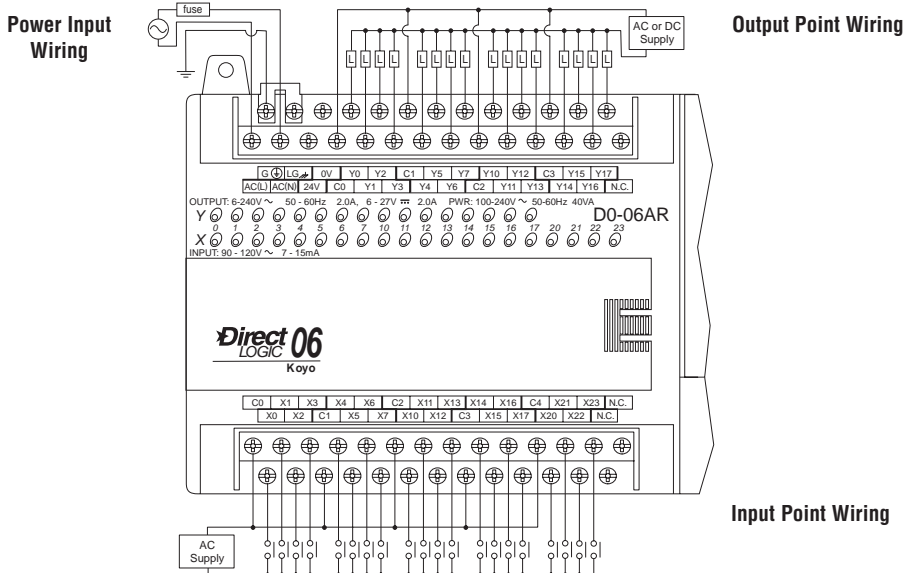
AC Input Specifications	
<b>Input Voltage Range (Min. - Max.)</b>	80 – 132 VAC, 47 - 63 Hz
<b>Operating Voltage Range</b>	90 – 120 VAC, 47 - 63 Hz
<b>Input Current</b>	8 mA @100 VAC at 50 Hz 10 mA @100 VAC at 60 Hz
<b>Max. Input Current</b>	12 mA @132 VAC at 50 Hz 15 mA @132 VAC at 60 Hz
<b>Input Impedance</b>	14K $\Omega$ @50 Hz, 12K $\Omega$ @60Hz
<b>ON Current/Voltage</b>	> 6 mA @ 75 VAC
<b>OFF Current/Voltage</b>	< 2 mA @ 20 VAC
<b>OFF to ON Response</b>	< 40 mS
<b>ON to OFF Response</b>	< 40 mS
<b>Status Indicators</b>	Logic Side
<b>Commons</b>	4 channels / common x 5 banks (isolated)

AC Output Specifications	
<b>Output Voltage Range (Min. - Max.)</b>	15 – 264 VAC, 47 – 63 Hz
<b>Operating Voltage</b>	17 – 240 VAC, 47 – 63 Hz
<b>On Voltage Drop</b>	1.5 VAC (>50mA) 4.0 VAC (<50mA)
<b>Max Current</b>	0.5 A / point, 1.5 A / common
<b>Max leakage current</b>	<4 mA @ 264 VAC
<b>Max inrush current</b>	10 A for 10 mS
<b>Minimum Load</b>	10 mA
<b>OFF to ON Response</b>	1 mS
<b>ON to OFF Response</b>	1 mS +1/2 cycle
<b>Status Indicators</b>	Logic Side
<b>Commons</b>	4channels / common x 4 banks (isolated)
<b>Fuses</b>	None (external recommended)

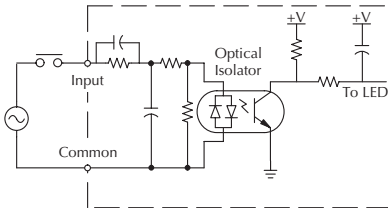
## D0-06AR I/O Wiring Diagram

The D0-06AR PLC has twenty AC inputs and sixteen relay contact outputs. The following diagram shows a typical field wiring example. The AC external power connection uses four terminals at the left as shown.

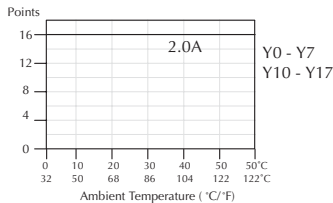
The twenty AC input channels use terminals on the bottom of the connector. Inputs are organized into five banks of four. Each bank has a common terminal. The wiring example below shows all commons connected together, but separate supplies and common circuits may be used. The equivalent input circuit shows one channel of a typical bank.



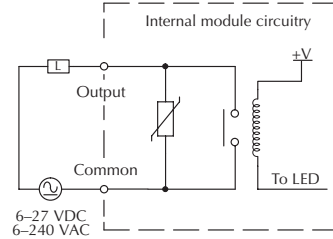
Equivalent Input Circuit



Derating Chart for Relay Outputs



Equivalent Output Circuit



The sixteen relay output channels use terminals on the right side top connector. Outputs are organized into two banks of three normally-open relay contacts. Each bank has a common terminal. The wiring example on the last page shows all commons connected together, but separate supplies and common circuits may be used. The equivalent output circuit shows one channel of a typical bank. The relay contacts can switch AC or DC voltages.

D0-06AR General Specifications	
<b>External Power Requirements</b>	100 – 240 VAC, 40 VA maximum,
<b>Communication Port 1 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity</b>	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
<b>Communication Port 2 9600 baud (default), 8 data bits, 1 stop bit, odd parity</b>	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / print, ASCII in/out
<b>Programming cable type</b>	D2-DSCBL
<b>Operating Temperature</b>	32 to 131° F (0 to 55 C)
<b>Storage Temperature</b>	-4 to 158° F (-20 to 70 C)
<b>Relative Humidity</b>	5 to 95% (non-condensing)
<b>Environmental air</b>	No corrosive gases permitted
<b>Vibration</b>	MIL STD 810C 514.2
<b>Shock</b>	MIL STD 810C 516.2
<b>Noise Immunity</b>	NEMA ICS3-304
<b>Terminal Type</b>	Removable
<b>Wire Gauge</b>	One AWG16 or two AWG18, AWG24 minimum

AC Input Specifications X0-X23	
<b>Input Voltage Range (Min. - Max.)</b>	80 – 132 VAC, 47 - 63 Hz
<b>Operating Voltage Range</b>	90 – 120 VAC, 47 -63 Hz
<b>Input Current</b>	8 mA @ 100 VAC at 50 Hz 10 mA @ 100 VAC at 60 Hz
<b>Max. Input Current</b>	12 mA @ 132 VAC at 50 Hz 15 mA @ 132 VAC at 60 Hz
<b>Input Impedance</b>	14KΩ @50 Hz, 12KΩ @60 Hz
<b>ON Current/Voltage</b>	>6 mA @ 75 VAC
<b>OFF Current/Voltage</b>	<2 mA @ 20 VAC
<b>OFF to ON Response</b>	< 40 mS
<b>ON to OFF Response</b>	< 40 mS
<b>Status Indicators</b>	Logic Side
<b>Commons</b>	4 channels / common x 5 banks (isolated)

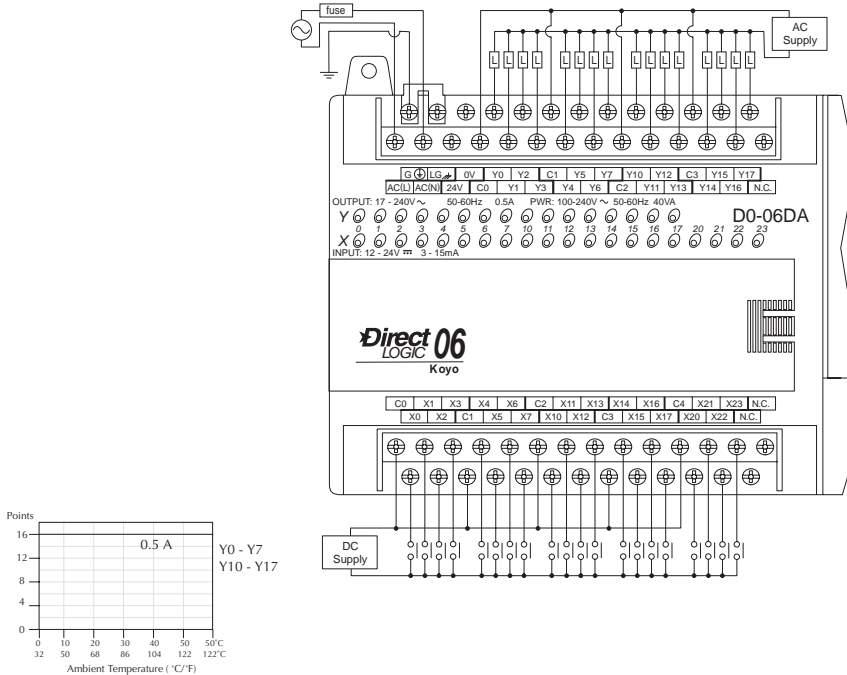
Relay Output Specifications Y0-Y17	
<b>Output Voltage Range</b>	(Min. – Max.) 5 – 264 VAC (47 -63 Hz), 5 – 30 VDC
<b>Operating Voltage Range</b>	6 – 240 VAC (47 -63 Hz), 6 – 27 VDC
<b>Output Current</b>	2A / point, 6A / common
<b>Max. leakage current</b>	0.1 mA @264VAC
<b>Smallest Recommended Load</b>	5 mA @5 VDC
<b>OFF to ON Response</b>	< 15 mS
<b>ON to OFF Response</b>	< 10 mS
<b>Status Indicators</b>	Logic Side
<b>Commons</b>	4 channels / common x 4 banks (isolated)
<b>Fuses</b>	None (external recommended)

## D0-06DA I/O Wiring Diagram

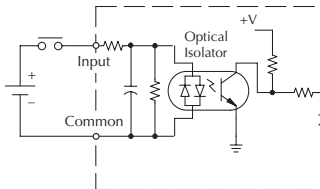
The D0-06DA PLC has twenty DC inputs and sixteen AC outputs. The following diagram shows a typical field wiring example. The AC external power connection uses four terminals as shown.

Inputs are organized into five banks of four. Each bank has an isolated common terminal, and may be wired as sinking or sourcing. The wiring example below shows all commons connected together, but separate supplies and common circuits may be used. The equivalent circuit for standard inputs is shown below, and the high-speed input circuit is shown to the left.

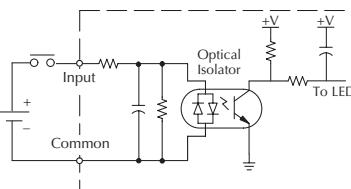
Outputs are organized into four banks of four triac switches. Each bank has a common terminal. The wiring example below shows all commons connected together, but separate supplies and common circuits may be used. The equivalent output circuit shows one channel of a typical bank.



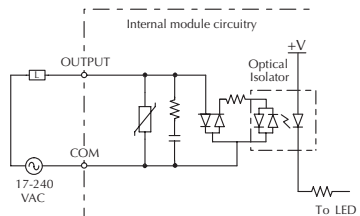
Derating Chart for AC Outputs



High Speed Inputs (X0-X3)



Standard Inputs (X4-X23)



Equivalent Output Circuit



D0-06DA General Specifications	
External Power Requirements	100 – 240 VAC, 40 VA maximum,
Communication Port 1 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
Communication Port 2 9600 baud (default), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence/print, ASCII in/out
Programming cable type	D2-DSCBL
Operating Temperature	32 to 131° F (0 to 55 C)
Storage Temperature	–4 to 158° F (–20 to 70 C)
Relative Humidity	5 to 95% (non-condensing)
Environmental air	No corrosive gases permitted
Vibration	MIL STD 810C 514.2
Shock	MIL STD 810C 516.2
Noise Immunity	NEMA ICS3–304
Terminal Type	Removable
Wire Gauge	One AWG16 or two AWG18, AWG24 minimum

DC Input Specifications		
Parameter	High-Speed Inputs, X0 – X3	Standard DC Inputs X4 – X23
Input Voltage Range	10.8 – 26.4 VDC	10.8 – 26.4 VDC
Operating Voltage Range	12 – 24 VDC	12 – 24 VDC
Maximum Voltage	30 VDC (7 kHz maximum frequency)	30 VDC
Minimum Pulse Width	70 µS	N/A
ON Voltage Level	> 10 VDC	> 10 VDC
OFF Voltage Level	< 2.0 VDC	< 2.0 VDC
Input Impedance	1.8 kΩ @ 12 – 24 VDC	2.8 kΩ @ 12 – 24 VDC
Minimum ON Current	>5 mA	>4 mA
Maximum OFF Current	< 0.5 mA	<0.5 mA
OFF to ON Response	<70 µS	2 – 8 mS, 4 mS typical
ON to OFF Response	<70 µS	2 – 8 mS, 4 mS typical
Status Indicators	Logic side	Logic side
Commons	4 channels / common x 5 bank (isolated)	

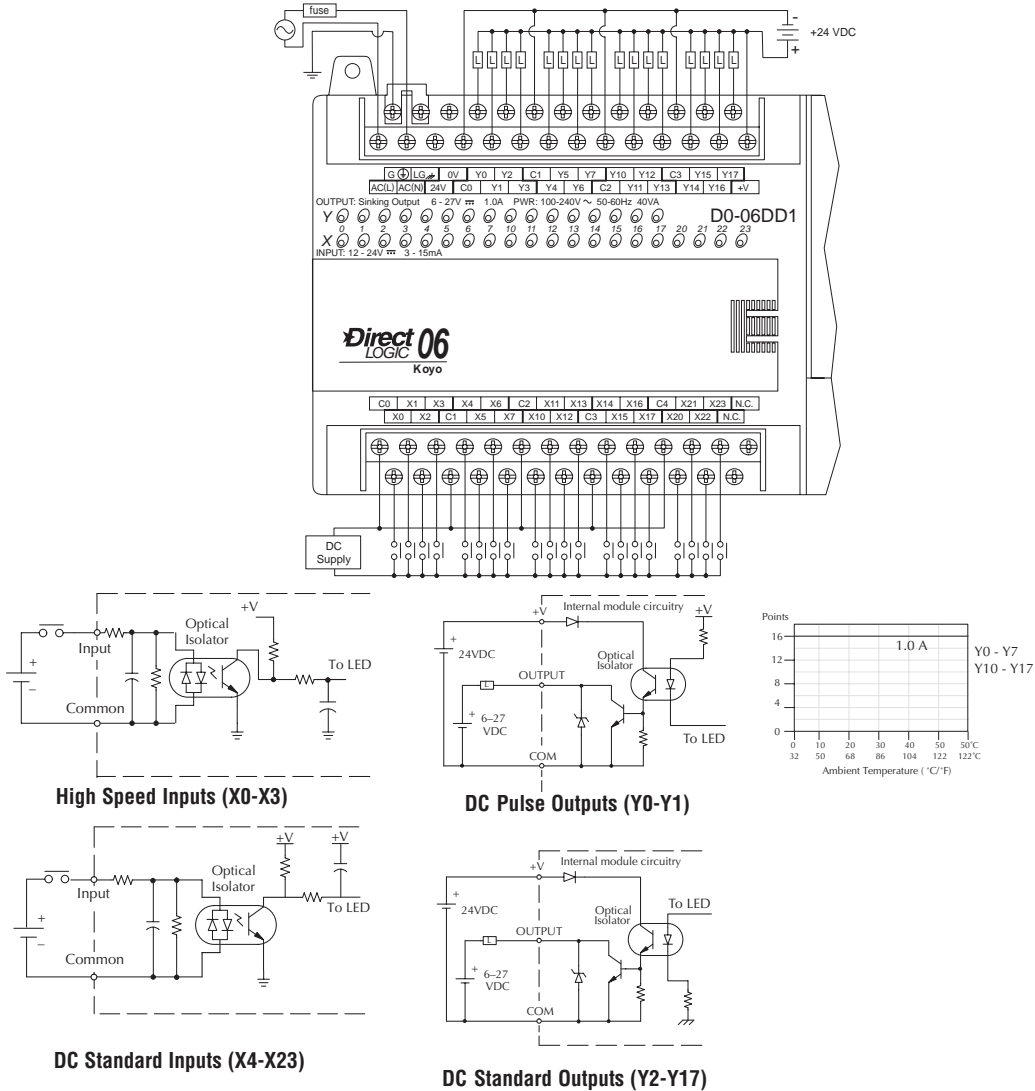
AC Output Specifications	
Output Voltage Range (Min. - Max.)	15 – 264 VAC, 47 – 63 Hz
Operating Voltage	17 – 240 VAC, 47 – 63 Hz
On Voltage Drop	1.5 VAC @> 50mA, 4 VAC @< 50mA
Max Current	0.5 A / point, 1.5 A / common
Max leakage current	< 4 mA @ 264 VAC, 60Hz
Max inrush current	10 A for 10 mS
Minimum Load	10 mA
OFF to ON Response	1 mS
ON to OFF Response	1 mS +1/2 cycle
Status Indicators	Logic Side
Commons	4 channels / common x 4 banks (isolated)
Fuses	None (external recommended)

## D0-06DD1 I/O Wiring Diagram

The D0-06DD1 PLC has twenty DC inputs and sixteen DC outputs. The following diagram shows a typical field wiring example. The AC external power connection uses four terminals as shown.

Inputs are organized into five banks of four. Each bank has an isolated common terminal, and may be wired as either sinking or sourcing inputs. The wiring example below shows all commons connected together, but separate supplies and common circuits may be used.

Outputs all share the same common. Note the requirement for external power.

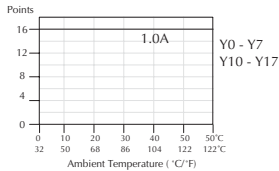
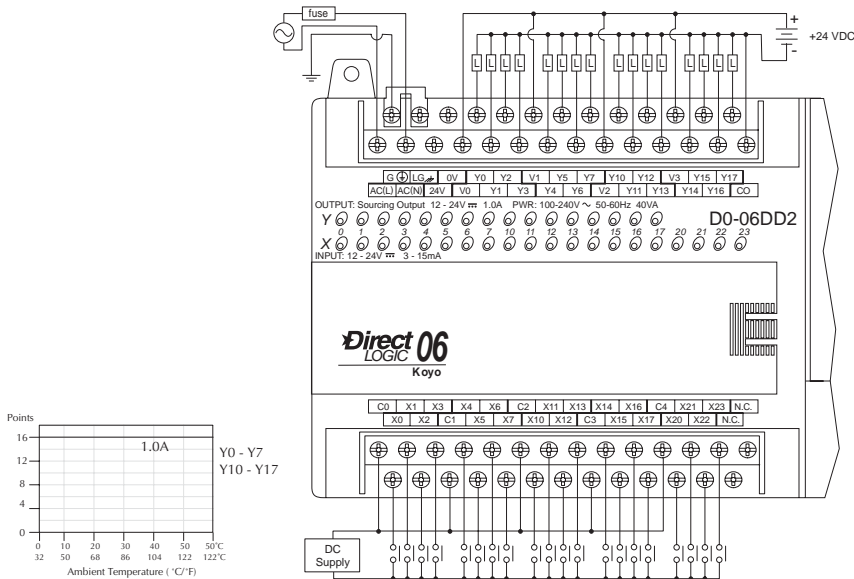


D0-06DD1 General Specifications		
External Power Requirements	100 – 240 VAC, 40 VA maximum,	
Communication Port 1 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)	
Communication Port 2 9600 baud (default), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / print, ASCII in/out	
Programming cable type	D2-DSCBL	
Operating Temperature	32 to 131° F (0 to 55 C)	
Storage Temperature	-4 to 158° F (-20 to 70 C)	
Relative Humidity	5 to 95% (non-condensing)	
Environmental air	No corrosive gases permitted	
Vibration	MIL STD 810C 514.2	
Shock	MIL STD 810C 516.2	
Noise Immunity	NEMA ICS3-304	
Terminal Type	Removable	
Wire Gauge	One AWG16 or two AWG18, AWG24 minimum	
DC Input Specifications		
Parameter	High-Speed Inputs, X0 – X3	Standard DC Inputs X4 – X23
Min. - Max. Voltage Range	10.8 – 26.4 VDC	10.8 – 26.4 VDC
Operating Voltage Range	12 – 24 VDC	12 – 24 VDC
Peak Voltage	30 VDC (7 kHz maximum frequency)	30 VDC
Minimum Pulse Width	100 µs	N/A
ON Voltage Level	> 10.0 VDC	> 10.0 VDC
OFF Voltage Level	< 2.0 VDC	< 2.0 VDC
Max. Input Current	6mA @12VDC, 13mA @24VDC	4mA @12VDC, 8.5mA @24VDC
Input Impedance	1.8 Ωk @ 12 – 24 VDC	2.8 Ωk @ 12 – 24 VDC
Minimum ON Current	>5 mA	>4 mA
Maximum OFF Current	< 0.5 mA	<0.5 mA
OFF to ON Response	<70 µS	2 – 8 mS, 4 mS typical
ON to OFF Response	<70 µS	2 – 8 mS, 4 mS typical
Status Indicators	Logic side	Logic side
Commons	4 channels / common x 5 banks isolated	
DC Output Specifications		
Parameter	Pulse Outputs Y0 – Y1	Standard Outputs Y2 – Y17
Min. - Max. Voltage Range	5 – 30 VDC	5 – 30 VDC
Operating Voltage	6 – 27 VDC	6 – 27 VDC
Peak Voltage	< 50 VDC (10 kHz max. frequency)	< 50 VDC
On Voltage Drop	0.3 VDC @ 1 A	0.3 VDC @ 1 A
Max Current (resistive)	0.5 A / pt., 1A / pt. as standard pt.	1.0 A / point
Max leakage current	15µA @ 30 VDC	15µA @ 30 VDC
Max inrush current	2 A for 100 mS	2 A for 100 mS
External DC power required	20 - 28 VDC Max 150mA	20 - 28 VDC Max 280mA (Aux. 24VDC powers V+ terminal (sinking outputs))
OFF to ON Response	< 10µ s	< 10 µs
ON to OFF Response	< 20 µs	< 60 µs
Status Indicators	Logic Side	Logic Side
Commons	4 channels / common x 4 banks non-isolated	
Fuses	None (external recommended)	

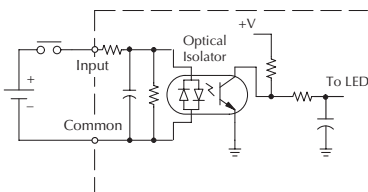
## D0-06DD2 I/O Wiring Diagram

The D0-06DD2 PLC has twenty DC inputs and sixteen sourcing DC outputs. The following diagram shows a typical field wiring example. The AC external power connection uses four terminals as shown.

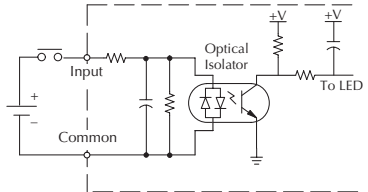
Inputs are organized into four banks of four. Each bank has an isolated common terminal, and may be wired as either sinking or sourcing inputs. The wiring example below shows all commons connected together, but separate supplies and common circuits may be used.



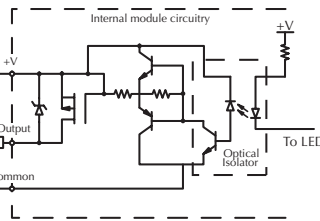
Derating Chart for DC Outputs



High Speed Inputs (X0-X3)

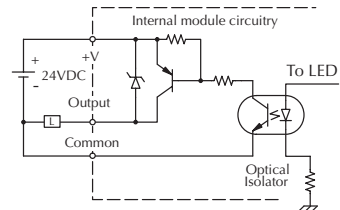


DC Standard Inputs (X4-X23)



DC Pulse Outputs (Y0-Y1)

All outputs share the same common. Note the requirement for external power.



DC Standard Outputs (Y2-Y17)

D0-06DD2 General Specifications		
External Power Requirements	100 – 240 VAC, 40 VA maximum,	
Communication Port 1 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)	
Communication Port 2 9600 baud (default), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence / print, ASCII in/out	
Programming cable type	D2-DSCBL	
Operating Temperature	32 to 131° F (0 to 55 C)	
Storage Temperature	-4 to 158° F (-20 to 70 C)	
Relative Humidity	5 to 95% (non-condensing)	
Environmental air	No corrosive gases permitted	
Vibration	MIL STD 810C 514.2	
Shock	MIL STD 810C 516.2	
Noise Immunity	NEMA ICS3-304	
Terminal Type	Removable	
Wire Gauge	One AWG16 or two AWG18, AWG24 minimum	
DC Input Specifications		
Parameter	High-Speed Inputs, X0 – X3	Standard DC Inputs X4 – X23
Min. - Max. Voltage Range	10.8 – 26.4 VDC	10.8 – 26.4 VDC
Operating Voltage Range	12 – 24 VDC	12 – 24 VDC
Peak Voltage	30 VDC (7 kHz maximum frequency)	30 VDC
Minimum Pulse Width	70 µs	N/A
ON Voltage Level	> 10.0 VDC	> 10.0 VDC
OFF Voltage Level	< 2.0 VDC	< 2.0 VDC
Max. Input Current	6mA @12VDC, 13mA @24VDC	4mA @12VDC, 8.5mA @24VDC
Input Impedance	1.8 Ωk @ 12 – 24 VDC	2.8 Ωk @ 12 – 24 VDC
Minimum ON Current	>5 mA	>4 mA
Maximum OFF Current	< 0.5 mA	<0.5 mA
OFF to ON Response	<70 µS	2 – 8 mS, 4 mS typical
ON to OFF Response	<70 µS	2 – 8 mS, 4 mS typical
Status Indicators	Logic side	Logic side
Commons	4 channels/common x 5 banks (isolated)	
DC Output Specifications		
Parameter	Pulse Outputs Y0 – Y1	Standard Outputs Y2 – Y17
Min. - Max. Voltage Range	10.8 -26.4 VDC	10.8 -26.4 VDC
Operating Voltage	12-24 VDC	12-24 VDC
Peak Voltage	< 50 VDC (10 kHz max. frequency)	< 50 VDC
On Voltage Drop	0.5VDC @ 1 A	1.2 VDC @ 1 A
Max Current (resistive)	0.5 A / pt., 1A / pt. as standard pt.	1.0 A / point
Max leakage current	15 µA @ 30 VDC	15 µA @ 30 VDC
Max inrush current	2 A for 100 mS	2 A for 100 mS
External DC power required	n/a	n/a
OFF to ON Response	< 10µs	< 10 µs
ON to OFF Response	< 20 µs	< 0.5 µs
Status Indicators	Logic Side	Logic Side
Commons	4 channels / common x 4 banks (non-isolated)	
Fuses	None (external recommended)	

D0-06DR I/O Wiring Diagram

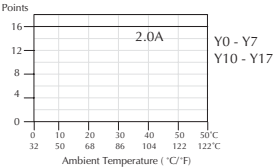
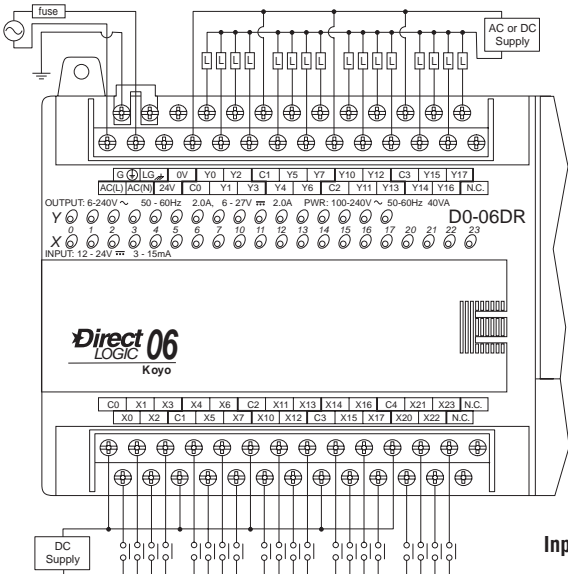
The D0-06DR PLCs feature twenty DC inputs and sixteen relay contact outputs. The following diagram shows a typical field wiring example. The AC external power connection uses four terminals as shown.

Inputs are organized into five banks of four. Each bank has an isolated common terminal, and may be wired as either sinking or sourcing inputs. The wiring example below shows all commons connected together, but separate supplies and common circuits may be used. The equivalent circuit for standard inputs is shown below, and the high-speed input circuit is shown to the left.

Outputs are organized into four banks of four normally-open relay contacts. Each bank has a common terminal. The wiring example below shows all commons connected together, but separate supplies and common circuits may be used. The equivalent output circuit shows one channel of a typical bank. The relay contacts can switch AC or DC voltages.

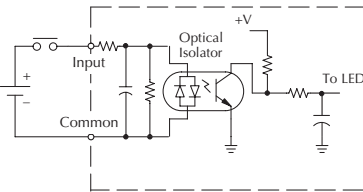
Power Input Wiring

Output Point Wiring

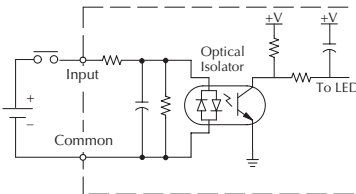


Derating Chart for Relay Outputs

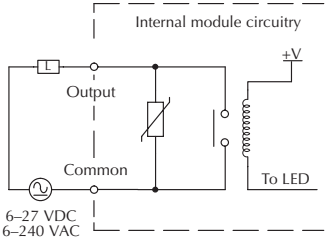
Equivalent Circuit, High-speed Inputs (X0-X3)



Equivalent Circuit, Standard Inputs (X4-X23)



Equivalent Output Circuit



D0-06DR General Specifications	
<b>External Power Requirements</b>	100 – 240 VAC, 40 VA maximum,
<b>Communication Port 1 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity</b>	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
<b>Communication Port 2 9600 baud (default), 8 data bits, 1 stop bit, odd parity</b>	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence /print, ASCII in/out
<b>Programming cable type</b>	D2-DSCBL
<b>Operating Temperature</b>	32 to 131° F (0 to 55 C)
<b>Storage Temperature</b>	-4 to 158° F (-20 to 70 C)
<b>Relative Humidity</b>	5 to 95% (non-condensing)
<b>Environmental air</b>	No corrosive gases permitted
<b>Vibration</b>	MIL STD 810C 514.2
<b>Shock</b>	MIL STD 810C 516.2
<b>Noise Immunity</b>	NEMA ICS3-304
<b>Terminal Type</b>	Removable
<b>Wire Gauge</b>	One AWG16 or two AWG18, AWG24 minimum

DC Input Specifications		
Parameter	High-Speed Inputs, X0 – X3	Standard DC Inputs X4 – X23
<b>Min. - Max. Voltage Range</b>	10.8 – 26.4 VDC	10.8 – 26.4 VDC
<b>Operating Voltage Range</b>	12 -24 VDC	12 -24 VDC
<b>Peak Voltage</b>	30 VDC (7 kHz maximum frequency)	30 VDC
<b>Minimum Pulse Width</b>	70 µs	N/A
<b>ON Voltage Level</b>	> 10 VDC	> 10 VDC
<b>OFF Voltage Level</b>	< 2.0 VDC	< 2.0 VDC
<b>Input Impedance</b>	1.8 kΩ @ 12 – 24 VDC	2.8 kΩ @ 12 – 24 VDC
<b>Max. Input Current</b>	6mA @12VDC 13mA @24VDC	4mA @12VDC 8.5mA @24VDC
<b>Minimum ON Current</b>	>5 mA	>4 mA
<b>Maximum OFF Current</b>	< 0.5 mA	<0.5 mA
<b>OFF to ON Response</b>	<70 µs	2 – 8 mS, 4 mS typical
<b>ON to OFF Response</b>	<70 µs	2 – 8 mS, 4 mS typical
<b>Status Indicators</b>	Logic side	Logic side
<b>Commons</b>	4 channels / common x 5 banks (isolated)	

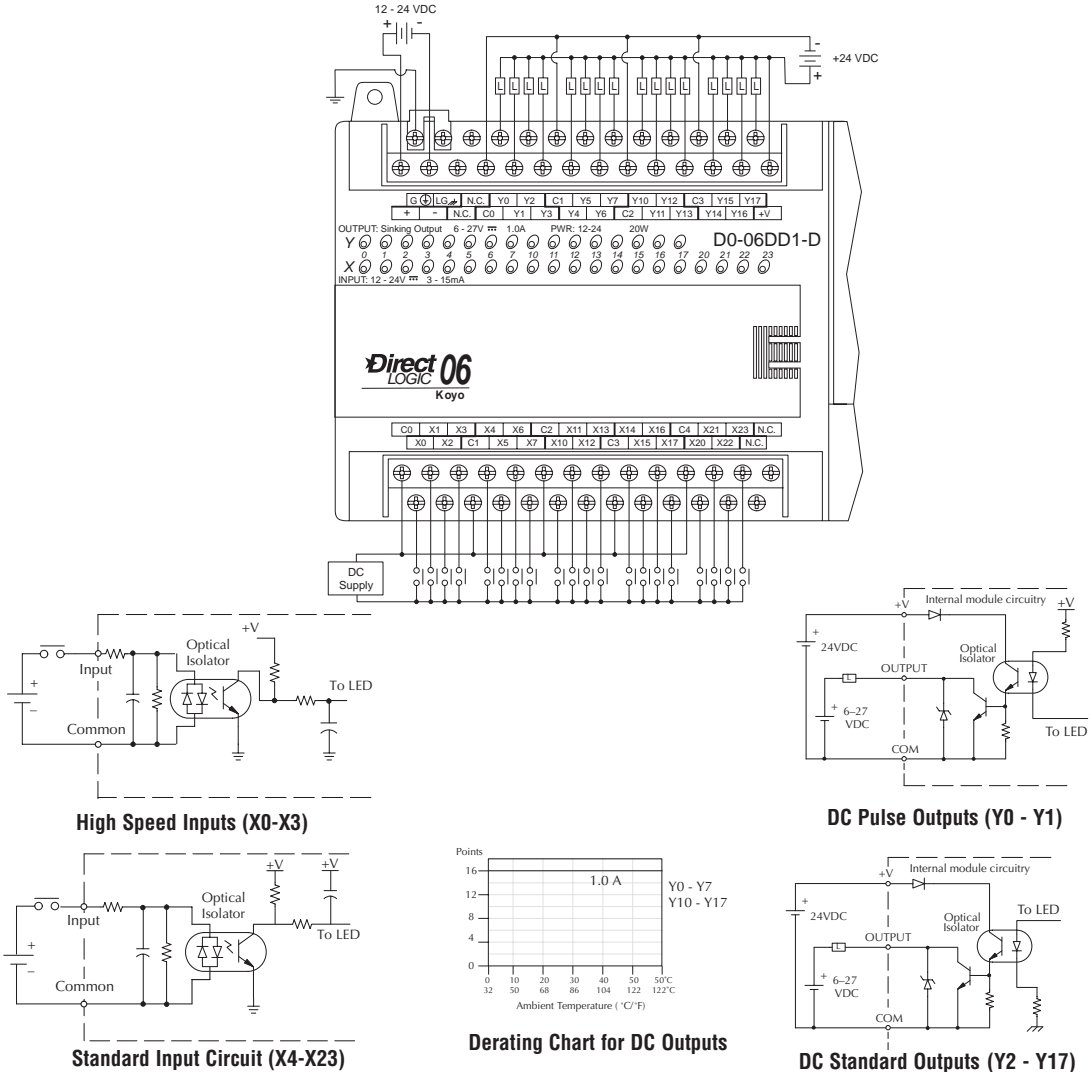
Relay Output Specifications	
<b>Output Voltage Range (Min. - Max.)</b>	5 -264 VAC (47 -63 Hz), 5 - 30 VDC
<b>Operating Voltage</b>	6 -240 VAC (47 -63 Hz), 6 - 27 VDC
<b>Output Current</b>	2A / point 6A / common
<b>Maximum Voltage</b>	264 VAC, 30 VDC
<b>Max leakage current</b>	0.1 mA @264 VAC
<b>Smallest Recommended Load</b>	5 mA
<b>OFF to ON Response</b>	< 15 mS
<b>ON to OFF Response</b>	< 10 mS
<b>Status Indicators</b>	Logic Side
<b>Commons</b>	4 channels / common x 4 banks (isolated)
<b>Fuses</b>	None (external recommended)

## D0-06DD1-D I/O Wiring Diagram

These micro PLCs feature twenty DC inputs and sixteen sinking DC outputs. The following diagram shows a typical field wiring example. The DC external power connection uses four terminals at the left as shown.

Inputs are organized into five banks of four. Each bank has an isolated common terminal, and may be wired as either sinking or sourcing inputs. The wiring example below shows all commons connected together, but separate supplies and common circuits may be used.

All outputs actually share the same common. Note the requirement for external power.

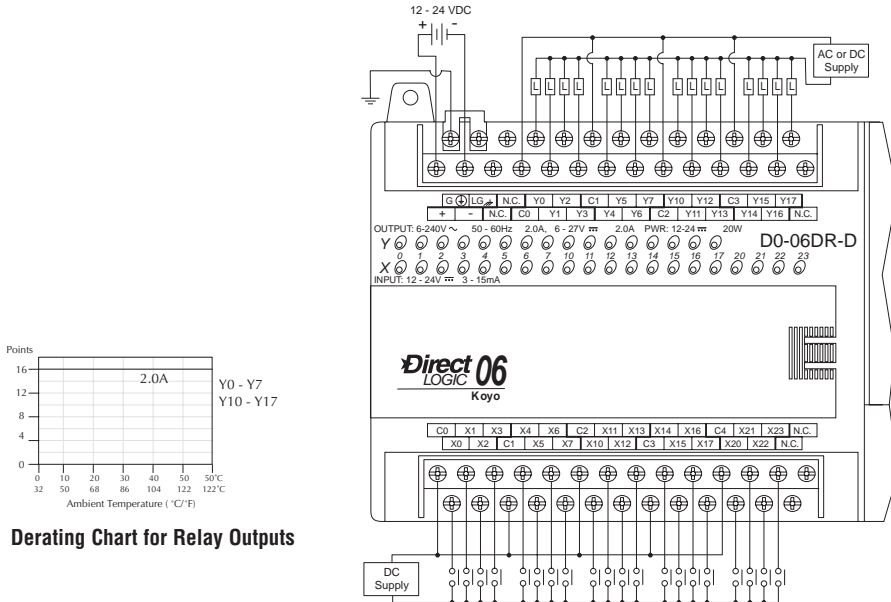




D0-06DD1-D General Specifications		
External Power Requirements	12 – 24 VDC, 20 W maximum,	
Communication Port 1: 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity	K–Sequence (Slave), DirectNET (Slave), MODBUS (Slave)	
Communication Port 2: 9600 baud (default), 8 data bits, 1 stop bit,odd parity	K–Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence/print, ASCII in/out	
Programming cable type	D2–DSCBL	
Operating Temperature	32 to 131° F (0 to 55 C)	
Storage Temperature	–4 to 158° F (–20 to 70 C)	
Relative Humidity	5 to 95% (non-condensing)	
Environmental air	No corrosive gases permitted	
Vibration	MIL STD 810C 514.2	
Shock	MIL STD 810C 516.2	
Noise Immunity	NEMA ICS3–304	
Terminal Type	Removable	
Wire Gauge	One AWG16 or two AWG18, AWG24 minimum	
DC Input Specifications		
Parameter	High–Speed Inputs, X0 – X3	Standard DC Inputs X4 – X23
Min. - Max. Voltage Range	10.8 – 26.4 VDC	10.8 – 26.4 VDC
Operating Voltage Range	12 – 24 VDC	12 – 24 VDC
Peak Voltage	30 VDC (7 kHz maximum frequency)	30 VDC
Minimum Pulse Width	70 µs	N/A
ON Voltage Level	>10.0 VDC	> 10.0 VDC
OFF Voltage Level	< 2.0 VDC	< 2.0 VDC
Max. Input Current	6mA @12VDC, 13mA @24VDC	4mA @12VDC, 8.5mA @24VDC
Input Impedance	1.8 kΩ @ 12 – 24 VDC	2.8 kΩ @ 12 – 24 VDC
Minimum ON Current	>5 mA	>4 mA
Maximum OFF Current	< 0.5 mA	<0.5 mA
OFF to ON Response	<70 µS	2 – 8 mS, 4 mS typical
ON to OFF Response	<70 µS	2 – 8 mS, 4 mS typical
Status Indicators	Logic side	Logic side
Commons	4 channels / common x 5 banks (isolated)	
DC Output Specifications		
Parameter	Pulse Outputs, Y0 – Y1	Standard Outputs, Y2 – Y17
Min. - Max. Voltage Range	5 – 30 VDC	5 – 30 VDC
Operating Voltage	6 – 27 VDC	6 – 27 VDC
Peak Voltage	< 50 VDC (10 kHz max. frequency)	< 50 VDC
On Voltage Drop	0.3 VDC @ 1 A	0.3 VDC @ 1 A
Max Current (resistive)	0.5 A / pt., 1A / pt. as standard pt.	1.0 A / point
Max leakage current	15 µA @ 30 VDC	15 µA @ 30 VDC
Max inrush current	2 A for 100	mS 2 A for 100 mS
External DC power required	20 - 28 VDC Max 150mA	20 - 28 VDC Max 150mA
OFF to ON Response	< 10 µs	< 10 µs
ON to OFF Response	< 20 µs	< 60 µs
Status Indicators	Logic Side	Logic Side
Commons	4 channels / common x 4 banks (non-isolated)	
Fuses	None (external recommended)	

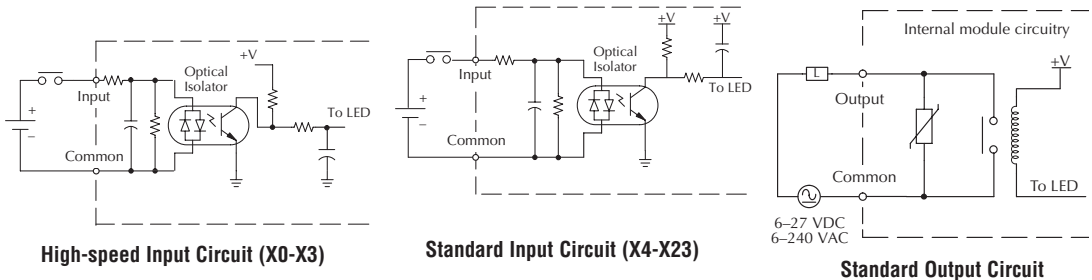
## D0-06DR-D I/O Wiring Diagram

The D0-06DR-D PLC has twenty DC inputs and sixteen relay contact outputs. The following diagram shows a typical field wiring example. The DC external power connection uses three terminals as shown.



Inputs are organized into five banks of four. Each bank has an isolated common terminal, and may be wired as either sinking or sourcing inputs. The wiring example above shows all commons connected together, but separate supplies and common circuits may be used.

Outputs are organized into four banks of four normally-open relay contacts. Each bank has a common terminal. The wiring example above shows all commons connected together, but separate supplies and common circuits may be used. The equivalent output circuit shows one channel of a typical bank. The relay contacts can switch AC or DC voltages.



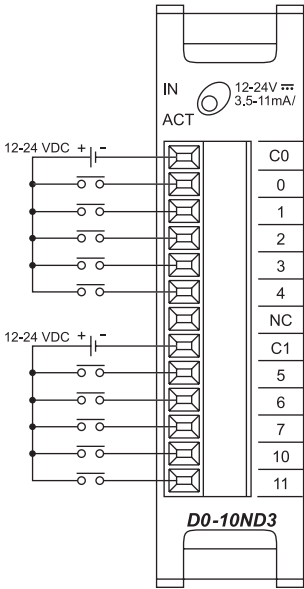
D0-06DR-D General Specifications	
<b>External Power Requirements</b>	12 – 24 VDC, 20 W maximum,
<b>Communication Port 1 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity</b>	K-Sequence (Slave), DirectNET (Slave), MODBUS (Slave)
<b>Communication Port 2 9600 baud (default), 8 data bits, 1 stop bit, odd parity</b>	K-Sequence (Slave), DirectNET (Master/Slave), MODBUS (Master/Slave), Non-sequence/print, ASCII in/out
<b>Programming cable type</b>	D2-DSCBL
<b>Operating Temperature</b>	32 to 131° F (0 to 55 C)
<b>Storage Temperature</b>	–4 to 158° F (–20 to 70 C)
<b>Relative Humidity</b>	5 to 95% (non-condensing)
<b>Environmental air</b>	No corrosive gases permitted
<b>Vibration</b>	MIL STD 810C 514.2
<b>Shock</b>	MIL STD 810C 516.2
<b>Noise Immunity</b>	NEMA ICS3–304
<b>Terminal Type</b>	Removable
<b>Wire Gauge</b>	One AWG16 or two AWG18, AWG24 minimum

DC Input Specifications		
Parameter	High-Speed Inputs, X0 – X3	Standard DC Inputs X4 – X23
<b>Min. - Max. Voltage Range</b>	10.8 – 26.4 VDC	10.8 – 26.4 VDC
<b>Operating Voltage Range</b>	12 -24 VDC	12 -24 VDC
<b>Peak Voltage</b>	30 VDC (7 kHz maximum frequency)	30 VDC
<b>Minimum Pulse Width</b>	70 µs	N/A
<b>ON Voltage Level</b>	> 10 VDC	> 10 VDC
<b>OFF Voltage Level</b>	< 2.0 VDC	< 2.0 VDC
<b>Input Impedance</b>	1.8 kΩ @ 12 – 24 VDC	2.8 kΩ @ 12 – 24 VDC
<b>Max. Input Current</b>	6mA @12VDC 13mA @24VDC	4mA @12VDC 8.5mA @24VDC
<b>Minimum ON Current</b>	>5 mA	>4 mA
<b>Maximum OFF Current</b>	< 0.5 mA	<0.5 mA
<b>OFF to ON Response</b>	<70 µs	2 – 8 mS, 4 mS typical
<b>ON to OFF Response</b>	< 70 µs	2 – 8 mS, 4 mS typical
<b>Status Indicators</b>	Logic side	Logic side
<b>Commons</b>	4 channels / common x 5 banks (isolated)	

Relay Output Specifications	
<b>Output Voltage Range (Min. - Max.)</b>	5 -264 VAC (47 -63 Hz), 5 - 30 VDC
<b>Operating Voltage</b>	6 -240 VAC (47 -63 Hz), 6 - 27 VDC
<b>Output Current</b>	2A / point 6A / common
<b>Maximum Voltage</b>	264 VAC, 30 VDC
<b>Max leakage current</b>	0.1 mA @264 VAC
<b>Smallest Recommended Load</b>	5 mA
<b>OFF to ON Response</b>	< 15 mS
<b>ON to OFF Response</b>	< 10 mS
<b>Status Indicators</b>	Logic Side
<b>Commons</b>	3 channels / common x 2 banks
<b>Fuses</b>	None (external recommended)

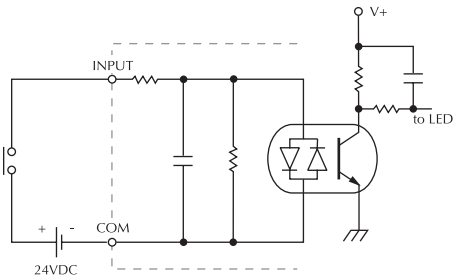
D0-10ND3  
10-point DC input module

D0-10ND3 Specifications	
Number of Inputs	10 (sink/source)
Input Voltage Range	10.8-26.4VDC
Operating Voltage Range	12-24VDC
Peak Voltage	30.0VDC
Input Current	Typical: 4.0mA @ 12VDC 8.5mA @ 24VDC
Maximum Input Current	11mA @ 26.4VDC
Input Impedance	2.8K $\Omega$ @ 12-24VDC
On Voltage Level	> 10.0 VDC
Off Voltage Level	< 2.0 VDC
Minimum ON Current	3.5mA
Minimum OFF Current	0.5mA
Off to On Response	2-8ms, Typ. 4ms
On to Off Response	2-8ms, Typ. 4ms
Status Indicators	Module activity: one green LED
Commons	2 non-isolated
Fuse	No fuse
Base Power Required	Typical. 35mA (all pts. ON)

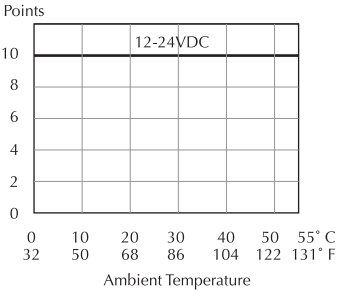


*Note: The DL06 must have firmware version V4.10 (or later) for this module to function properly.*

Equivalent input circuit

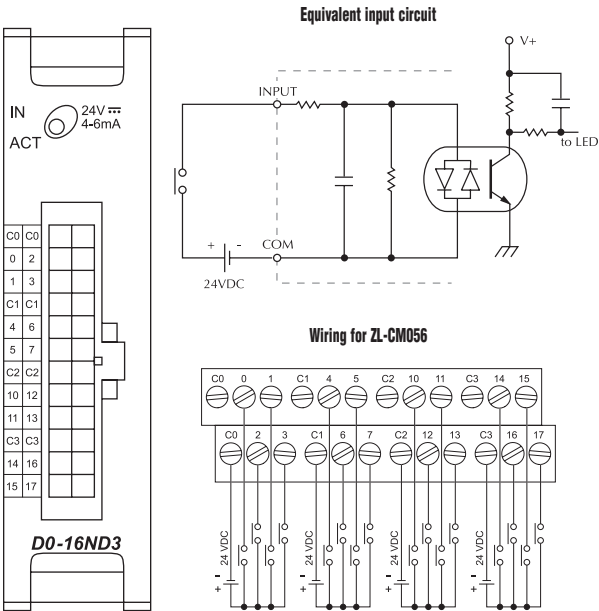


Derating chart

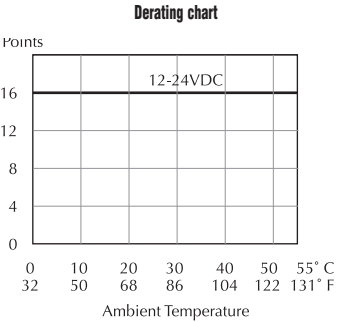


# D0-16ND316-point DC input module

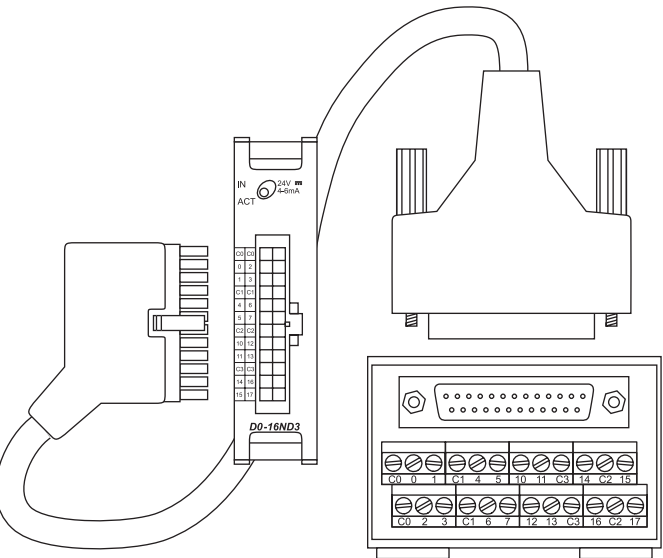
D0-16ND3 Specifications	
Number of Inputs	16 (sink/source)
Input Voltage Range	20-28VDC
Operating Voltage Range	24VDC
Peak Voltage	30.0VDC
Input Current	Typical: 4.0mA @ 24VDC
Maximum Input Current	6mA @ 28VDC
Input Impedance	4.7KΩ @ 24VDC
On Voltage Level	> 19.0 VDC
Off Voltage Level	< 7.0 VDC
Minimum ON Current	3.5mA
Minimum OFF Current	1.5mA
Off to on Response	2-8ms, Typ. 4ms
On to off Response	2-8ms, Typ. 4ms
Status Indicators	Module activity: one green LED
Commons	4 non-isolated
Fuse	No fuse
External DC power required	20-28VDC max. 200 mA (all pts. ON)
Base Power Required (5V)	Typical. 35mA (all pts. ON)



*Note: The DL06 must have firmware version V4.10 (or later) for this module to function properly.*



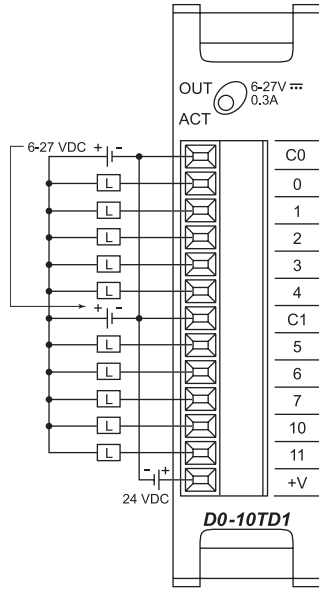
**Use ZipLink ZL-CBL056 cable and ZL-CM056 connector module or build your own cables using 24-pin Molex Micro Fit 3.0 receptacle, part number 43025, or compatible.**



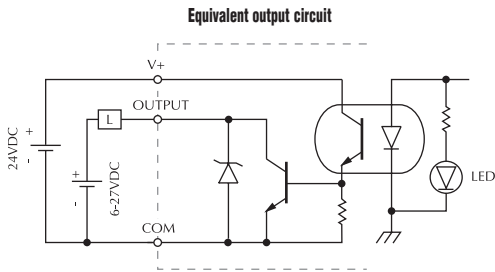
D0-10TD1

10-point DC output module

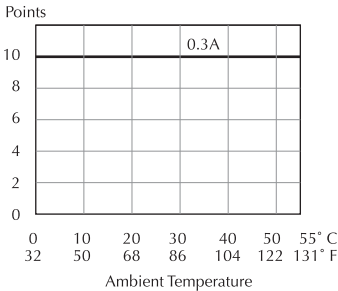
D0-10TD1 Specifications	
Number of Outputs	10 (sinking)
Operating Voltage Range	6-27VDC
Output Voltage Range	5-30VDC
Peak Voltage	50.0VDC
Maximum Output Current	0.3A/point 1.5A/common
Minimum Output Current	0.5mA
ON Voltage Drop	0.5.VDC @0.3A
Maximum Leakage Current	15µA @ 30.0VDC
Maximum Inrush Current	1A for 10ms
OFF to ON Response	<10µs
ON to OFF Response	<60µs
Status Indicators	Module activity: one green LED
Commons	2 non-isolated (5 points/common)
Fuse	No fuse
Base Power Required (5V)	Max. 150mA (All pts. on)



*Note: The DL06 must have firmware version V4.10 (or later) for this module to function properly.*



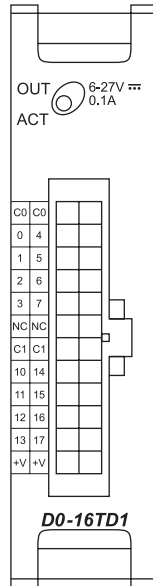
Derating chart



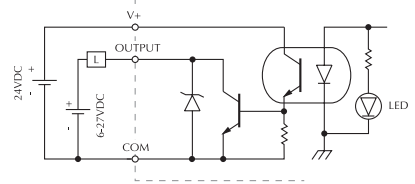
Use ZipLink ZL-CBL056 cable and ZL-CM056 connector module  
or build your own cables using 24-pin Molex Micro Fit 3.0 receptacle, part number 43025, or compatible.

## D0-16TD1 16-point DC output module

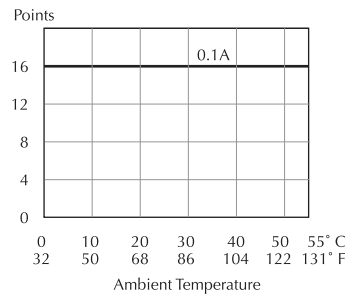
D0-16TD1 Specifications	
Number of Outputs	16 (sinking)
Operating Voltage Range	6-27VDC
Output Voltage Range	5-30VDC
Peak Voltage	50.0VDC
Maximum Output Current	0.1A/point 0.8A/common
Minimum Output Current	0.5mA
ON Voltage Drop	0.5.VDC @0.1A
Maximum Leakage Current	15µA @ 30.0VDC
Maximum Inrush Current	1A for 10ms
OFF to ON Response	<0.5 ms
ON to OFF Response	<0.5 ms
Status Indicators	Module activity: one green LED
Commons	2 isolated (8 points/common)
Fuse	No fuse
External DC power required	20-28 VDC max 70 mA (all pts. ON)
Base Power Required (5V)	Max. 200mA (All pts. ON)



Equivalent input circuit

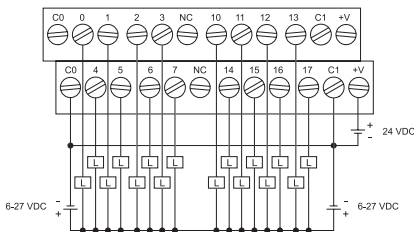


Derating chart

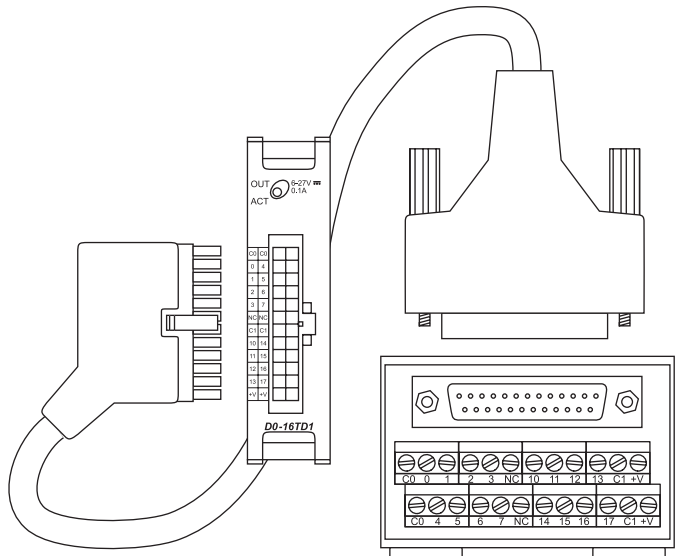


*Note: The DL06 must have firmware version V4.10 (or later) for this module to function properly.*

Wiring for ZL-CM056



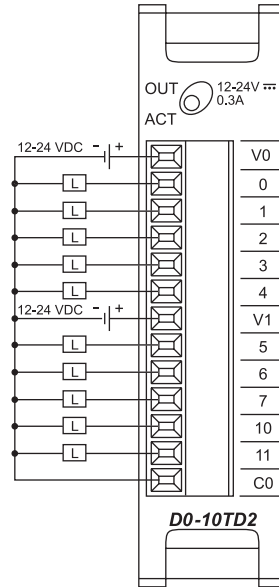
**Use ZipLink ZL-CBL056 cable and ZL-CM056 connector module or build your own cables using 24-pin Molex Micro Fit 3.0 receptacle, part number 43025, or compatible.**



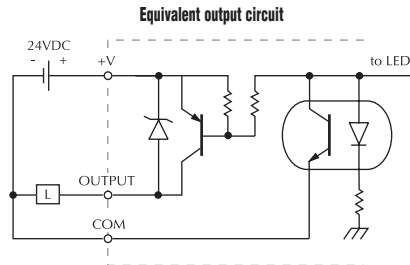
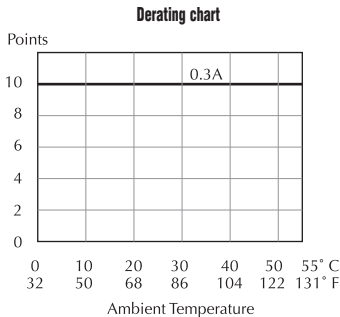
## D0-10TD2

### 10-point DC output module

D0-10TD2 Output Specifications	
Number of Outputs	10 (sourcing)
Operating Voltage Range	12-24VDC
Output Voltage Range	10.8-26.4VDC
Peak Voltage	50.0VDC
Maximum Output Current	0.3A/point 1.5A/common
Minimum Output Current	0.5mA
ON Voltage Drop	1.0.VDC @0.3A
Maximum Leakage Current	1.5µA @ 30.0VDC
Maximum Inrush Current	1A for 10ms
OFF to ON Response	<10µs
ON to OFF Response	<60µs
Status Indicators	Module activity: one green LED
Commons	2 non-isolated (5 points/common)
Fuse	No fuse
Base Power Required (5V)	Max. 150mA (All pts. On)



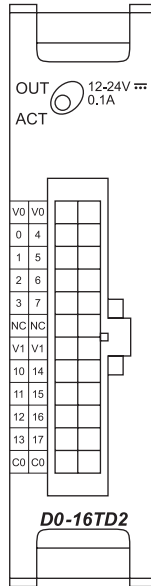
*Note: The DL06 must have firmware version V4.10 (or later) for this module to function properly.*



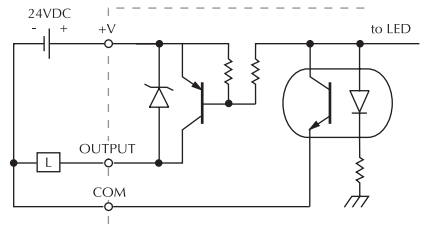


## D0-16TD2 16-point DC output module

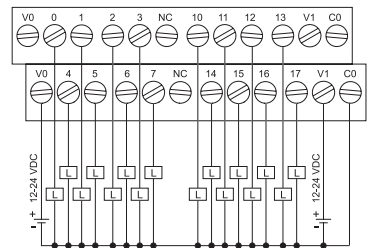
D0-16TD2 Specifications	
Number of Outputs	16 (sourcing)
Operating Voltage Range	12-24VDC
Output Voltage Range	10.8-26.4VDC
Peak Voltage	50.0VDC
Maximum Output Current	0.1A/point 0.8A/common
Minimum Output Current	0.5mA
ON Voltage Drop	1.0.VDC @ 0.1A
Maximum Leakage Current	1.5µA @ 26.4VDC
Maximum Inrush Current	1A for 10ms
OFF to ON Response	<0.5 ms
ON to OFF Response	<0.5 ms
Status Indicators	Module activity: one green LED
Commons	2 non-isolated (8 points/common)
Fuse	No fuse
Base Power Required (5V)	Max. 200mA (All pts. ON)



Equivalent output circuit

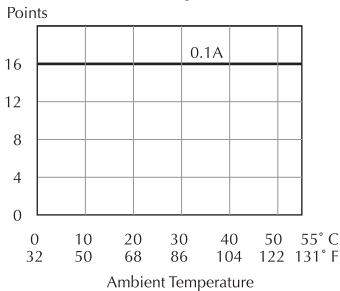


Wiring for ZL-CM056

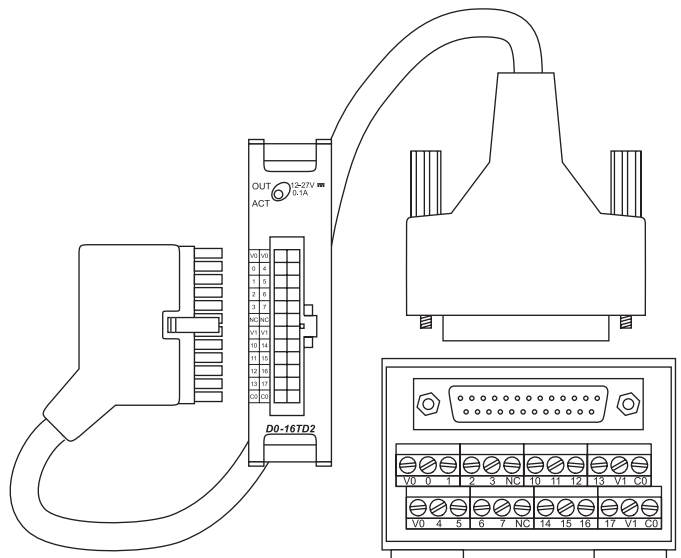


*Note: The DL06 must have firmware version V4.10 (or later) for this module to function properly.*

Derating chart



**Use ZipLink ZL-CBL056 cable and ZL-CM056 connector module or build your own cables using 24-pin Molex Micro Fit 3.0 receptacle, part number 43025, or compatible.**



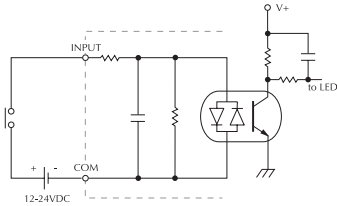
D0-07CDR 4-point DC input and 4-point relay output module

Input Specifications		Output Specifications	
Inputs per module	4 (sink/source)	Outputs per module	3
Operating voltage range	12-24 VDC	Operating voltage range	6-27 VDC/6-240 VAC
Input voltage range	10.8-26.4 VDC	Output type	Relay, form A, SPST
Peak voltage	30.0 VDC	Peak voltage	30.0 VDC/264 VAC
Maximum input current	11 mA @ 26.4 VDC	Maximum current (resistive)	1 A/point, 4 A/common
Input current	Typical: 4mA @ 12VDC 8.5 mA @ 24VDC	Minimum load current	5mA @ 5VDC
Input impedance	2.8k $\Omega$ @ 12-24VDC	Maximum leakage current	0.1 mA @ 264 VAC
ON voltage level	>10.0 VDC	ON voltage drop	N/A
OFF voltage level	< 2.0 VDC	Maximum inrush current	Output 3A for 10 ms Common: 10A for 10 ms
Minimum ON current	3.5 mA	ON to OFF response	< 10 ms
Maximum OFF current	0.5 mA	OFF to ON response	< 15 ms
ON to OFF response	2-8 ms, typical 4 ms	Status indicators	Module activity: one green LED
OFF to ON response	2-8 ms, typical 4 ms	Commons	1 (3 points/common)
Commons	1 (4 points/common)	Fuse	N/A
		Base power required (5V)	Max. 200 mA (all points ON)

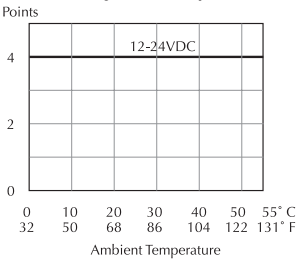


Note: The DL06 must have firmware version V4.10 (or later) for this module to function properly.

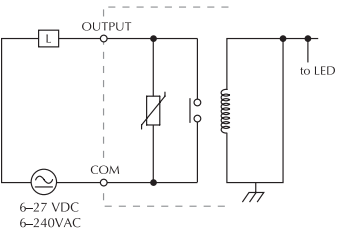
Equivalent input circuit



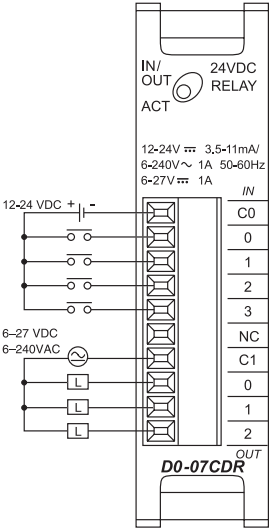
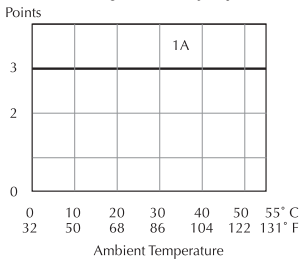
Derating chart for DC inputs



Equivalent output circuit

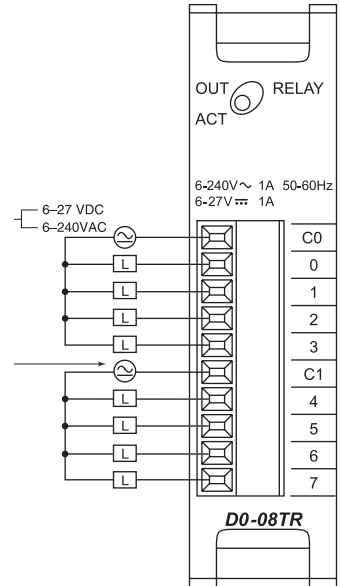


Derating chart for relay outputs



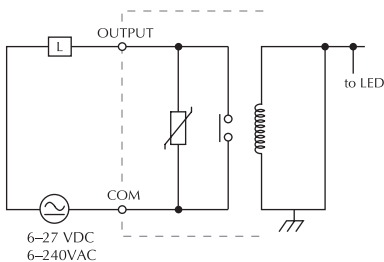
## D0-08TR 8-point Relay output module

D0-08TR Specifications	
Outputs per module	8
Operating voltage range	6-27 VDC/6-240 VAC
Output type	Relay, form A, SPST
Peak voltage	30.0 VDC/264 VAC
Maximum current (resistive)	1 A/point, 4 A/common
Minimum load current	0.5mA
Maximum leakage current	0.1 mA @ 264 VAC
ON voltage drop	N/A
Maximum inrush current	Output: 3A for 10 ms. Common: 10A for 10 ms
ON to OFF response	< 10 ms
OFF to ON response	< 15 ms
Status indicators	Module activity: one green LED
Commons	2 isolated. (4 points/common)
Fuse	N/A
Base power required (5V)	Max. 280 mA (all points ON)

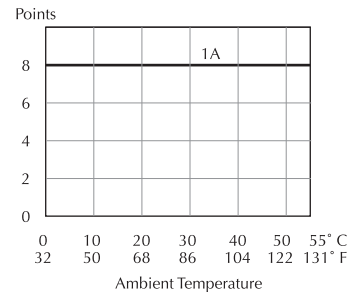


*Note: The DL06 must have firmware version V4.10 (or later) for this module to function properly.*

Equivalent output circuit



Derating chart



D0-08CDD1

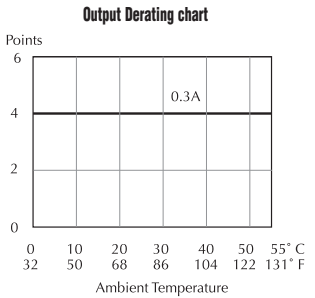
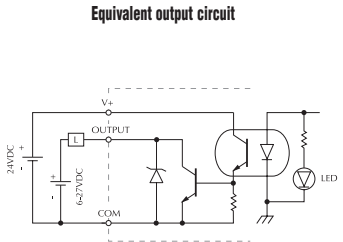
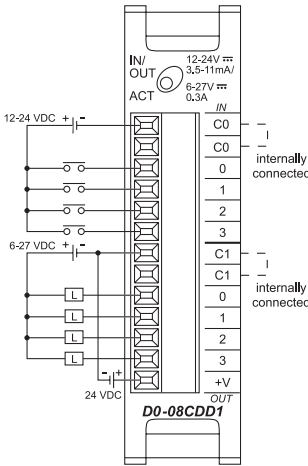
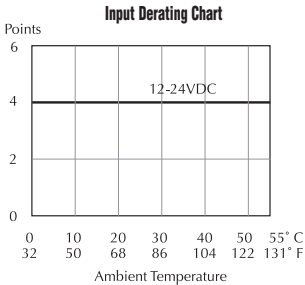
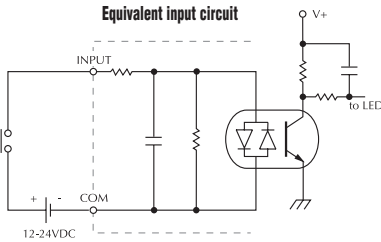
4-point DC input and 4-point relay output module

Input Specifications	
Inputs per module	4 (sink/source)
Operating voltage range	10.8-26.4 VDC
Input voltage range	12-24 VDC
Peak voltage	30.0 VDC
Maximum input current	11 mA @ 26.4 VDC
Input current	Typical: 4mA @ 12 VDC 8.5 mA @ 24 VDC
Input impedance	2.8KΩ @ 12-24 VDC
ON voltage level	>10.0 VDC
OFF voltage level	< 2.0 VDC
Minimum ON current	3.5 mA
Maximum OFF current	0.5 mA
ON to OFF response	2-8 ms, typical 4 ms
OFF to ON response	2-8 ms, typical 4 ms
Commons	2 non-isolated (4points/common)

Output Specifications	
Outputs per module	4 (sinking)
Operating voltage range	6-27 VDC
Output voltage range	5-30 VDC
Peak voltage	50.0 VDC/
Maximum output current	0.3 A/point, 1.2 A/common
Minimum output current	0.5 mA
Maximum leakage current	1.5 μA @ 30.0 VDC
ON voltage drop	0.5 VDC @ 0.3A
Maximum inrush current	1A for 10 ms
ON to OFF response	<60 ms
OFF to ON response	<10 ms
Status indicators	Module activity: one green LED
Commons	2 non-isolated (4 points/common)
Fuse	N/A
Base power required (5V)	Max. 200 mA (all points ON)
External DC power required (24V)	20 - 28 VDC, maximum 80 mA (all pts. ON)



Note: The DL06 must have firmware version V4.10 (or later) for this module to function properly.



## I/O Addressing

### Module I/O Points and Addressing

Each option module has a set number of I/O points. This holds true for both the discrete modules and the analog modules. The following chart shows the number of I/O points per module when used in the DL06.

DC Input Modules	I/O Points	Slot 1 I/O Address
D0-10ND3	10 Input	X100 - X107 and X110 - X111
D0-16ND	16 Input	X100 - X107 and X110 - X117
DC Output Modules	I/O Points	Slot 1 I/O Address
D0-10TD1	10 Output	Y100 - Y107 and Y110 - Y111
D0-16TD1	16 Output	Y100 - Y107 and Y110 - Y117
D0-10TD2	10 Output	Y100 - Y107 and Y110 - Y111
D0-16TD2	16 Output	Y100 - Y107 and Y110 - Y117
Relay Output Modules	I/O Points	Slot 1 I/O Address
D0-08TR	8 Output	Y100 - X107
Combination Modules	I/O Points	Slot 1 I/O Address
D0-07CDR	4 Input, 3 Output	X100 - X103 and Y100 - Y102
D0-08CDD1	4 Input, 4 Output	X100 - X103 and Y100 - Y103

# HIGH-SPEED INPUT AND PULSE OUTPUT FEATURES

---



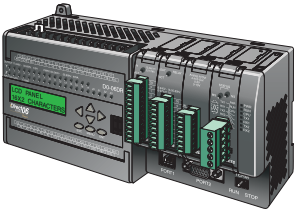
## In This Chapter

Introduction .....	3-2
Choosing the HSIO Operating Mode .....	3-4
Mode 10: High-Speed Counter .....	3-7
Mode 20: Up/Down Counter .....	3-24
Mode 30: Pulse Output .....	3-38
Mode 40: High-Speed Interrupts .....	3-64
Mode 50: Pulse Catch Input .....	3-69
Mode 60: Discrete Inputs with Filter .....	3-73

# Introduction

## Built-in Motion Control Solution

Many machine control applications require various types of simple high-speed monitoring and control. These applications usually involve some type of motion control, or high-speed interrupts for time-critical events. The DL06 Micro PLC solves this traditionally expensive problem with built-in CPU enhancements. Let's take a closer look at the available high-speed I/O features.



The available high-speed input features are:

- High Speed Counter (7 kHz max.) with up to 24 counter presets and built-in interrupt subroutine, counts up only, with reset
- Quadrature encoder inputs to measure counts and clockwise or counter clockwise direction (7 kHz max.), counts up or down, with reset
- High-speed interrupt inputs for immediate response to critical or time-sensitive tasks
- Pulse catch feature to monitor one input point, having a pulse width as small as 100µs (0.1ms)
- Programmable discrete filtering (both on and off delay up to 99ms) to ensure input signal integrity (this is the default mode for inputs X0–X3)

The available pulse output features are:

- Single-axis programmable pulse output (10 kHz max.) with three profile types, including trapezoidal moves, registration, and velocity control

## Availability of HSIO Features

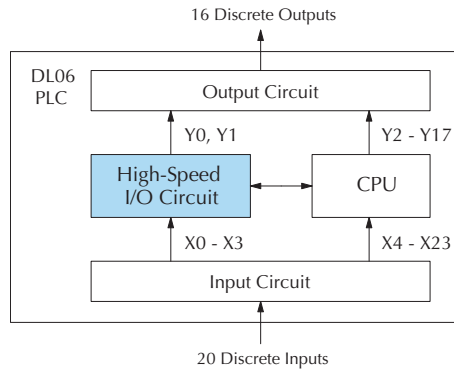
**IMPORTANT:** Please note the following restrictions on availability of features:

- High-speed input options are available only on DL06s with DC inputs.
- Pulse output options are available only on DL06s with DC outputs.
- Only one HSIO feature may be in use at one time. You cannot use a high-speed input feature and the pulse output at the same time.

Specifications				
DL06 Part Number	Discrete Input Type	Discrete Output Type	High-Speed Input	Pulse Output
DO-06AA	AC	AC	No	No
DO-06AR	AC	Relay	No	No
DO-06DA	DC	AC	Yes	No
DO-06DD1	DC	DC	Yes	Yes
DO-06DD2	DC	DC	Yes	Yes
DO-06DR	DC	Relay	Yes	No
DO-06DD1-D	DC	DC	Yes	Yes
DO-06DR-D	DC	Relay	Yes	No

## Dedicated High-Speed I/O Circuit

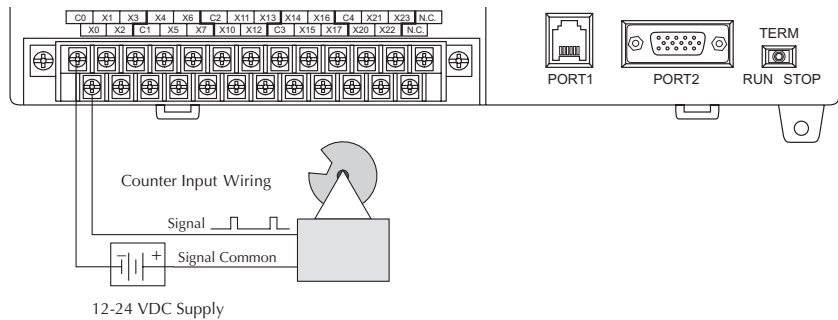
The internal CPU's main task is to execute the ladder program and read/write all I/O points during each scan. In order to service high-speed I/O events, the DL06 includes a special circuit which is dedicated to a portion of the I/O points. Refer to the DL06 block diagram in the figure below.



The high-speed I/O circuit (HSIO) is dedicated to the first four inputs (X0 – X3) and the first two outputs (Y0 – Y1). We might think of this as a “CPU helper”. In the default operation (called “Mode 60”) the HSIO circuit just passes through the I/O signals to or from the CPU, so that all twenty inputs behave equally and all sixteen outputs behave equally. When the CPU is configured in any other HSIO Mode, the HSIO circuit imposes a specialized function on the portion of inputs and outputs shown. The HSIO circuit *operates independently of the CPU program scan*. This provides accurate measurement and capturing of high-speed I/O activity while the CPU is busy with ladder program execution.

## Wiring Diagrams for Each HSIO Mode

After choosing the appropriate HSIO mode for your application, you'll need to refer to the section in this chapter for that specific mode. Each section includes wiring diagrams to help you connect the High-Speed I/O points correctly to field devices. An example of the quadrature counter mode diagram is shown below.





# Choosing the HSIO Operating Mode

## Understanding the Six Modes

The High-Speed I/O circuit operates in one of 6 basic modes as listed in the table below. The number in the left column is the mode number (later, we'll use these numbers to configure the PLC). Choose one of the following modes according to the primary function you want from the dedicated High-Speed I/O circuit. You can simply use all twenty inputs and sixteen outputs as regular I/O points with Mode 60.

High Speed I/O Basic Modes		
Mode		Mode Features
10	High-Speed Counter	Two 7 kHz counters with 24 presets and reset input, counts up only, cause interrupt on preset
20	Up/Down Counter	7 kHz up/down counter with 24 presets and reset, causes interrupt on preset
		Channel A / Channel B 7 kHz quadrature input, counts up and down
30	Pulse Output	Stepper control – pulse and direction signals, programmable motion profile (10kHz max.)
40	High-Speed Interrupt	Generates an interrupt based on input transition or time
50	Pulse Catch	Captures narrow pulses on a selected input
60	Filtered Input	Rejects narrow pulses on selected inputs

In choosing one of the six high-speed I/O modes, the I/O points listed in the table below operate only as the function listed. If an input point is not specifically used to support a particular mode, it usually operates as a filtered input by default. Similarly, output points operate normally unless Pulse Output mode is selected.

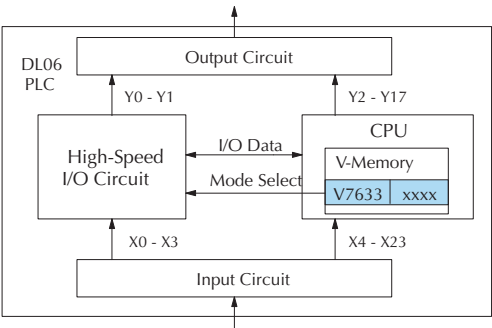
Physical I/O Point Usage							
Mode		DC Input Points				DC Output Points	
		X0	X1	X2	X3	Y0	Y1
10	High-Speed Counter	Counter #1	Counter #2, Interrupt, Pulse Input or Filtered Input	Reset #1, Interrupt, Pulse Input or Filtered Input	Reset #2, Interrupt, Pulse Input or Filtered Input	Regular Output	Regular Output
20	Up/Down counter (Standard counting)	Up Counting	Down Counting	Reset, Pulse Input or Filtered Input	Pulse Input or Filtered Input	Regular Output	Regular Output
	Up/Down counter (Quadrature counting)	Phase A Input	Phase B Input				
30	Pulse Output	Pulse Input or Filtered Input	Pulse Input or Filtered Input	Pulse Input or Filtered Input	Pulse Input or Filtered Input	Pulse or CW Pulse	Direction or CCW Pulse
40	High-Speed Interrupt	Interrupt	Interrupt, Pulse Input or Filtered Input	Interrupt, Pulse Input or Filtered Input	Interrupt, Pulse Input or Filtered Input	Regular Output	Regular Output
50	Pulse Catch	Pulse Input	Pulse Input, Interrupt or Filtered Input	Pulse Input, Interrupt or Filtered Input	Pulse Input, Interrupt or Filtered Input	Regular Output	Regular Output
60	Filtered Input	Filtered Input	Filtered Input	Filtered Input	Filtered Input	Regular Output	Regular Output

### Default Mode

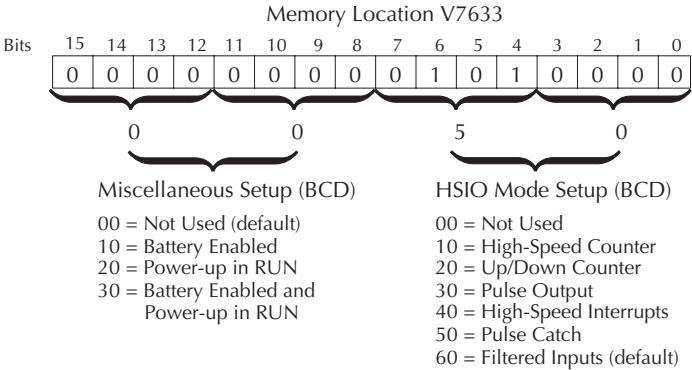
Mode 60 (Filtered Inputs) is the default mode. The DL06 is initialized to this mode at the factory, and any time you initialize the scratchpad memory. In the default condition, X0–X3 are filtered inputs (10 mS delay) and Y0–Y1 are standard outputs.

Configuring the HSIO Mode

If you have chosen a mode suited to the high-speed I/O needs of your application, we're ready to proceed to configure the PLC to operate accordingly. In the block diagram below, notice the V-memory detail in the expanded CPU block. V-memory location V7633 determines the functional mode of the high-speed I/O circuit. *This is the most important V-memory configuration value for HSIO functions!*



The contents of V7633 is a 16-bit word, to be entered in binary-coded decimal. The figure below defines what each 4-bit BCD digit of the word represents.



Bits 0 – 7 define the mode number 00, 10.. 60 previously referenced in this chapter. The example data “2050” shown selects Mode 50 – Pulse Catch (BCD = 50).

Configuring Inputs X0 – X3

In addition to configuring V7633 for the HSIO mode, you'll need to program the next four locations in certain modes according to the desired function of input points X0 – X3. Other memory locations may require configuring, depending on the HSIO mode (see the corresponding section for particular HSIO modes).

	V-Memory	
Mode	V7633	xxxx
X0	V7634	xxxx
X1	V7635	xxxx
X2	V7636	xxxx
X3	V7637	xxxx

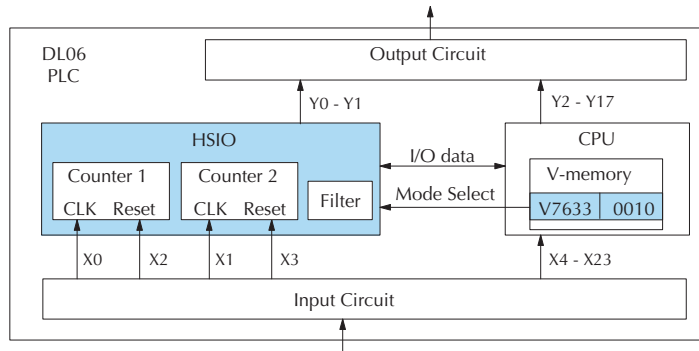
## Mode 10: High-Speed Counter

### Purpose

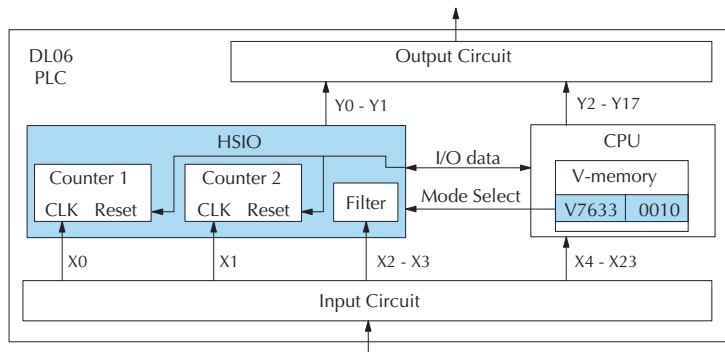
The HSIO circuit contains two high-speed counters. A single pulse train from an external source (X0) clocks the counter on each signal leading edge. The counter counts only upwards, from 0 to 99999999. The counter compares the current count with up to 24 preset values, which you define. The purpose of the presets is to quickly cause an action upon arrival at specific counts, making it ideal for such applications as cut-to-length. It uses counter registers CT174 to CT177 in the CPU.

### Functional Block Diagram

Refer to the block diagram below. When the lower byte of HSIO Mode register V7633 contains a BCD “10”, the high-speed up counter in the HSIO circuit is enabled. X0 and X1 automatically become the “clock” inputs for the high-speed counters, incrementing them upon each off-to-on transition. The external reset input on X2 and X3 are the default configuration for Mode 10.

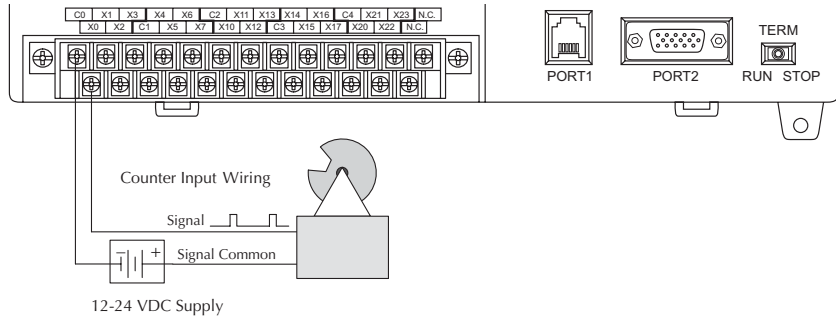


Instead of using X2 and X3 as dedicated reset inputs, you can configure X2 and X3 as normal filtered inputs. In this way, the counter reset must be generated in ladder logic.



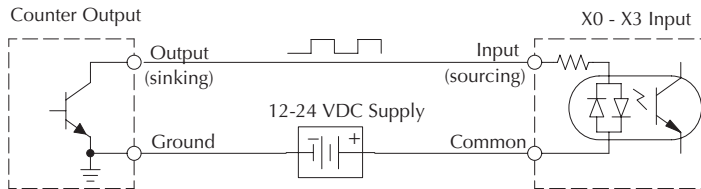
### Wiring Diagram

A general wiring diagram for counters/encoders to the DL06 in HSIO Mode 10 is shown below. Many types of pulse-generating devices may be used, such as proximity switches, single-channel encoders, magnetic or optical sensors, etc. Devices with sinking outputs (NPN open collector) are probably the best choice for interfacing. If the counter sources to the inputs, it must output 12 to 24 VDC. Note that devices with 5V sourcing outputs will not work with DL06 inputs.

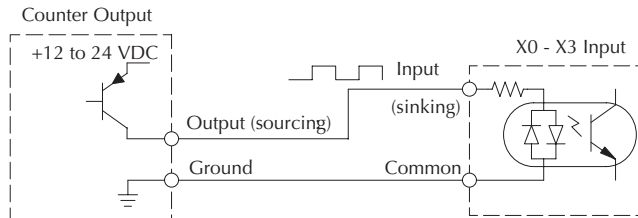


### Interfacing to Counter Inputs

The DL06's DC inputs are flexible in that they detect current flow in either direction, so they can be wired to a counter with either sourcing or sinking outputs. In the following circuit, a counter has open-collector NPN transistor outputs. It sinks current from the PLC input point, which sources current. The power supply can be the FA-24PS or another supply (+12VDC or +24VDC), as long as the input specifications are met.

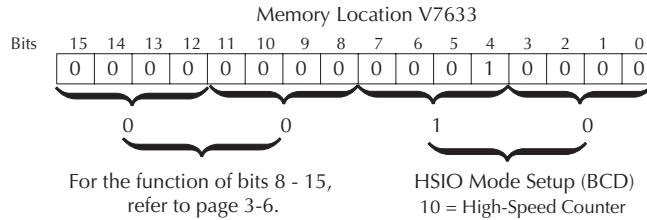


In the circuit diagram below, an encoder has open-emitter PNP transistor outputs. It sources current to the PLC input point, which sinks the current back to ground. Since the encoder sources current, no additional power supply is required. However, note that the encoder output must be 12 to 24 volts (5V encoder outputs will not work).



## Setup for Mode 10

V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 10 in the lower byte of V7633 to select the High-Speed Counter Mode.



Choose the most convenient method of programming V7633 from the following:

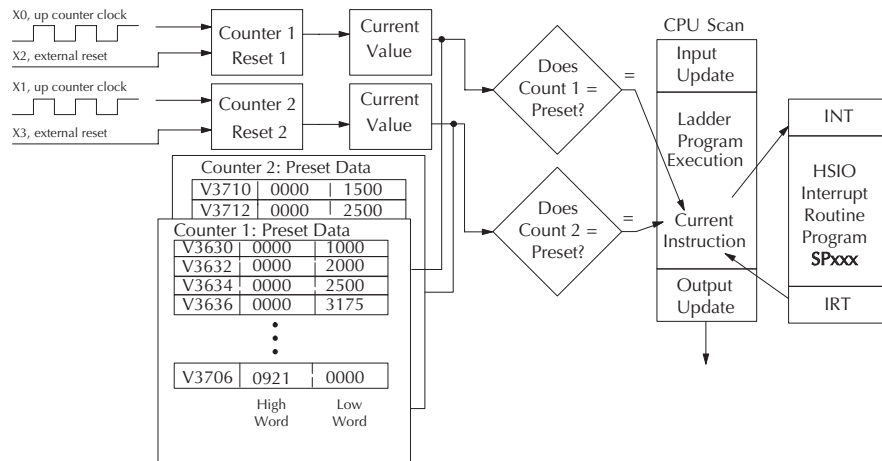
- Include load and out instructions in your ladder program
- **Direct**SOFT32's memory editor or Data View
- Use the Handheld Programmer D2-HPP

We recommend using the first method above so that the HSIO setup becomes an integral part of your application program. An example program later in this section shows how to do this.

## Presets and Special Relays

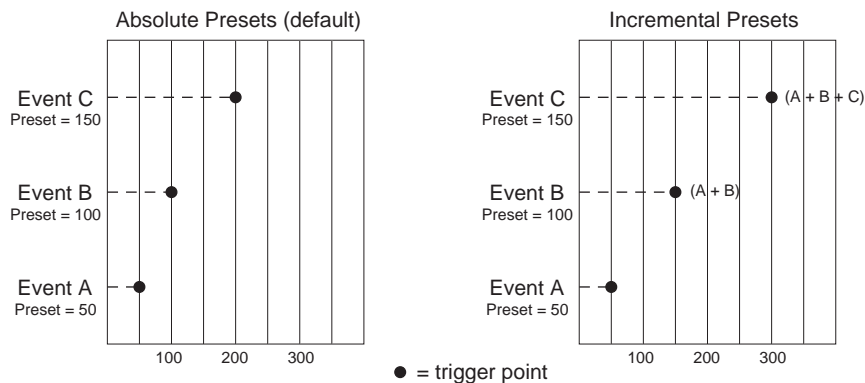
Presets are used to cause a particular action to occur when the count reaches the preset value. Refer to the figure below. Each counter features 24 presets, which you can program. Presets are double word numbers so they occupy two V-memory registers. The user selects the preset values, and the counter continuously compares the current count with the preset. When the two are equal, a special relay contact is energized and program execution jumps to the interrupt routine.

We recommend using the special relay(s) in the interrupt service routine to cause any immediate action you desire. After the interrupt service routine is complete, the CPU returns to the ladder program, resuming program execution from the point of interruption. The compare function is ready for the next preset event.



### Absolute and Incremental Presets

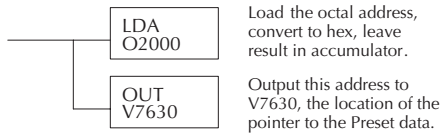
Two preset modes are available, absolute and incremental. Presets are entered into a contiguous block of V-memory registers. In the absolute mode, each preset is treated as the total count. In the incremental mode, the presets are cumulative. Incremental presets represent the number of counts between events.



In the example above, presets are established at 50, 100, and 150. The difference between absolute and incremental modes is shown. Absolute presets trigger events at the preset values, 50, 100, and 150. Incremental presets trigger events at the cumulative totals: 50, 150, and 300.

## Preset Data Starting Location

V7630 is the pointer to the V-memory location which contains the beginning of the Preset Data Tables. The default starting location for the Preset Data Tables is V3630 (default after initializing scratchpad). However, you may change this by programming a different value in V7630. Use the LDA and OUT instructions as shown:



Preset Table Pointer

V7630	2000
-------	------

Preset Table

V2001	V2000	0000	1000
V2003	V2002	0000	2000
V2005	V2004	0000	2500
V2007	V2006	0000	3175

⋮

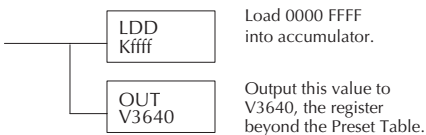
V2077	V2076	0000	0000
-------	-------	------	------

## Using Fewer than 24 Presets

When all 24 available presets are used, the CPU knows automatically when it reaches the end of the preset table. When using fewer than 24 presets, however, it is necessary to signal the CPU that it has reached the last preset. The way to signal the end of the block of presets is to insert one of the following “table-end” codes into the next available register pair:

Table-end Code	Applicable Mode	Meaning
0000 FFFF	Absolute and Incremental	Signals end of presets
0000 00FF	Incremental	Signals end of presets and restarts presets. Does not reset accumulated pulse counts of CT174 or CT176.
0000 FF00	Incremental	Signals end of presets, restarts presets and resets accumulated pulse counts of CT174 or CT176.

As shown in the table above, each of the “table-end” signals has a different meaning. Use the LDD Kffff instruction to insert the table-end code into the next register pair beyond the preset table. In the example, four presets are used. The 0000 FFFF in V3641-V3640 indicates the previous preset was the last preset.



Default Preset Table Example

V3631	V3630	0000	1000
V3633	V3632	0000	2000
V3635	V3634	0000	2500
V3637	V3636	0000	3175
V3641	V3640	0000	FFFF

In absolute mode, the counter and the cumulative total are reset each time a preset is reached. In incremental mode, you can choose not to reset the counter or the cumulative total, or you can choose to reset only the counter, or you can choose to reset the counter and the cumulative total when the table-end code is read. In the example, FFFF has been placed in V3640 since the last preset was in V3636, and we are using fewer than 24 presets.



**NOTE:** In incremental mode each successive preset must be greater than the previous preset value. If a preset value is less than a lower-numbered preset value, the CPU cannot compare to that value, since the counter can only count upwards.



### Equal Relay Numbers

The following table lists all 24 preset register default locations for each high-speed counter. Each occupies two 16-bit V-memory registers. The corresponding special relay contact number is in the next column. We might also call these “equal” relay contacts, because they are true (closed) when the present high-speed counter value is equal to the preset value. Each contact remains closed until the counter value equals the next preset value.

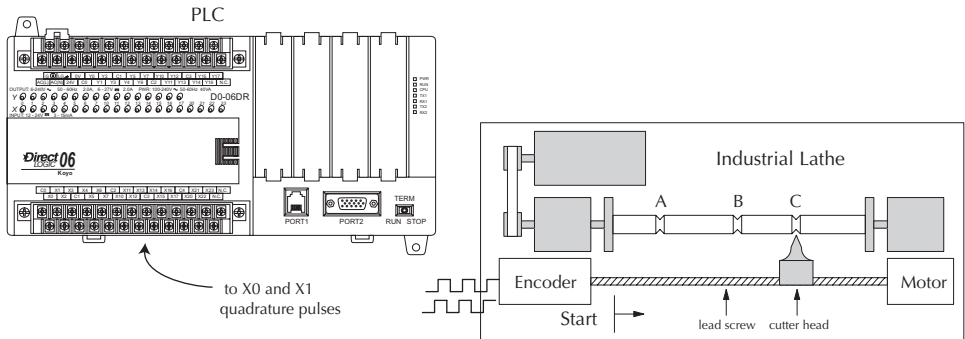
Preset Register Table					
Counter 1 Preset	Preset V-memory Register	Special Relay Number	Counter 2 Preset	Preset V-memory Register	Special Relay Number
1	V3631 / V3630	SP540	1	V3711/V3710	SP570
2	V3633 / V3632	SP541	2	V3713/V3712	SP571
3	V3635 / V3634	SP542	3	V3715/V3714	SP572
4	V3637 / V3636	SP543	4	V3717/V3716	SP573
5	V3641 / V3640	SP544	5	V3721/V3720	SP574
6	V3643 / V3642	SP545	6	V3723/V3722	SP575
7	V3645 / V3644	SP546	7	V3725/V3724	SP576
8	V3647 / V3646	SP547	8	V3727/V3726	SP577
9	V3651 / V3650	SP550	9	V3731/V3730	SP600
10	V3653 / V3652	SP551	10	V3733/V3732	SP601
11	V3655 / V3654	SP552	11	V3735/V3734	SP602
12	V3657 / V3656	SP553	12	V3737/V3736	SP603
13	V3661 / V3660	SP554	13	V3741/V3740	SP604
14	V3663 / V3662	SP555	14	V3743/V3742	SP605
15	V3665 / V3664	SP556	15	V3745/V3744	SP606
16	V3667 / V3666	SP557	16	V3747/V3746	SP607
17	V3671 / V3670	SP560	17	V3751/V3750	SP610
18	V3673 / V3672	SP561	18	V3753/V3752	SP611
19	V3675 / V3674	SP562	19	V3755/V3754	SP612
20	V3677 / V3676	SP563	20	V3757/V3756	SP613
21	V3701 / V3700	SP564	21	V3761/V3760	SP614
22	V3703 / V3702	SP565	22	V3763/V3762	SP615
23	V3705 / V3704	SP566	23	V3765/V3764	SP616
24	V3707 / V3706	SP567	24	V3767/V3766	SP617

The consecutive addresses shown above for each relay are those assigned by the CPU as default addresses. The Pointer for the start of these addresses is stored by the CPU at V7630. If you have a conflict of addresses because of pre-existing code written to these addresses, you can change the default block of addresses merely by having your ladder logic place a different pointer value in V7630. To change the table location, use the LDA and OUT instructions as shown on the previous page.

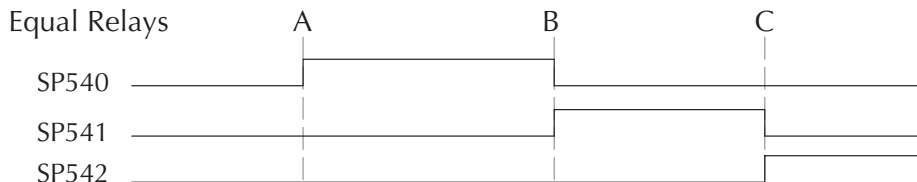
## Calculating Your Preset Values

The preset values occupy two data words each. They can range in value from -8388608 to 8388607, just like the high-speed counter value. All 24 values are absolute values, meaning that each one is an offset from the counter zero value.

The preset values must be individually derived for each application. In the industrial lathe diagram below, the PLC monitors the position of the lead screw by counting pulses. At points A, B, and C along the linear travel, the cutter head pushes into the work material and cuts a groove.



The timing diagram below shows the duration of each equal relay contact closure. Each contact remains on until the next one closes. All go off when the counter resets.



**NOTE:** Each successive preset must be two numbers greater than the previous preset value. In the industrial lathe example,  $B > A + 2$  and  $C > B + 2$ .

### X Input Configuration

The configurable discrete input options for High-Speed Counter Mode are listed in the table below. Input X0 is dedicated for the first counter clock input. Input X1 can be the clock for the second counter or a filtered input. The section on Mode 60 operation at the end of this chapter describes programming the filter time constants. Inputs X2 and X3 can be configured as the counter resets, with or without the interrupt option. The interrupt option allows the reset input (X2 and X3) to cause an interrupt like presets do, but there is no SP relay contact closure (instead, X2 and X3 will be on during the interrupt routine, for 1 scan). Or finally, X2 and X3 may be left simply as a filtered input.

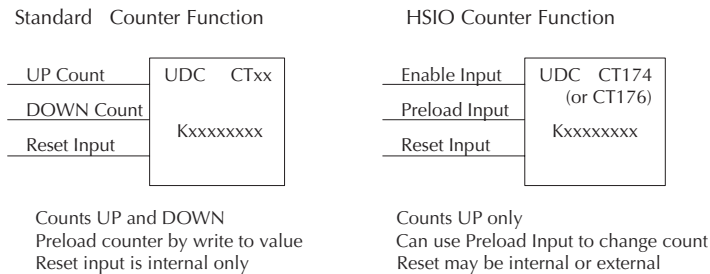
Input Options			
Input	Configuration Register	Function	Hex Code Required
X0	V7634	Counter #1 Clock	0001 (absolute) (default)
			0101 (incremental)
X1	V7635	Counter #2 Clock	0001 (absolute) (default)
			0101 (incremental)
		Interrupt	0004
		Pulse Input	0005
		Filtered Input	xx06, xx = filter time 0 - 99 ms (BCD)
X2	V7636	Counter #1 Reset (no interrupt)	0007* (default) 0207*
		Counter #1 Reset (with interrupt)	0107* 0307*
		Interrupt	0004
		Pulse Input	0005
		Filtered Input	xx06, xx= filter time 0 - 99 ms (BCD)
X3	V7637	Counter #2 Reset (no interrupt)	0007* (default) 0207*
		Counter #2 Reset (with interrupt)	0107* 0307*
		Interrupt	0004
		Pulse Input	0005
		Filtered Input	xx06, xx= filter time 0 - 99 ms (BCD)

\*With the counter reset, you have the option of a normal reset or a faster reset. However, the fast reset does not recognize changed preset values during program execution. When '0007' or '0107' are set in V7636 or V7637 and preset values are changed during program execution, the DL06 recognizes the changed preset values at the time of the reset. When '0207' or '0307' are set in V7636 or V7637 the CPU does not check for changed preset values, so the DL06 has a faster reset time.

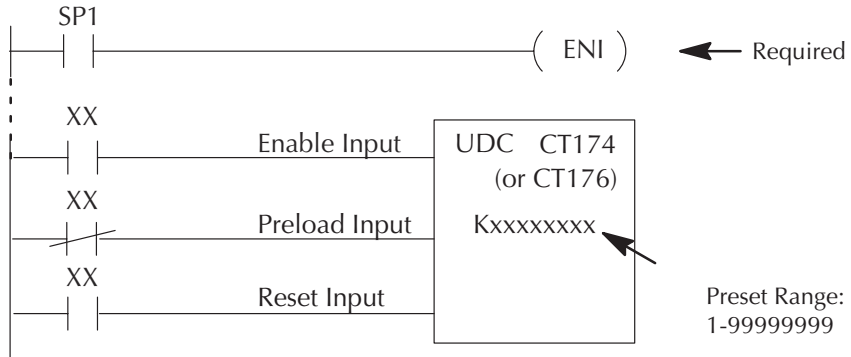
## Writing Your Control Program

The mnemonic for the counter instruction is UDC (up-down counter). The DL06 can have up to 128 counters, labeled CT0 through CT177. The high speed counter in the HSIO circuit is accessed in ladder logic by using UDC CT174 and CT176. It uses counter registers CT174 through CT177 exclusively when the HSIO mode 10 is active (otherwise, CT174 through CT177 are available for standard counter use). The HSIO counter needs two registers because it is a double-word counter. It has three inputs as shown. The first input (Enable) allows counting when active. The middle input is used to preload the counter value. The bottom signal is the reset. The Preload Input must be off while the counter is counting.

The next figure shows how the HSIO counter will appear in a ladder program. Note that the Enable Interrupt (ENI) command must execute before the counter value reaches the first preset value. We do this at powerup by using the first scan relay. When using the counter but not the presets and interrupt, we can omit the ENI.



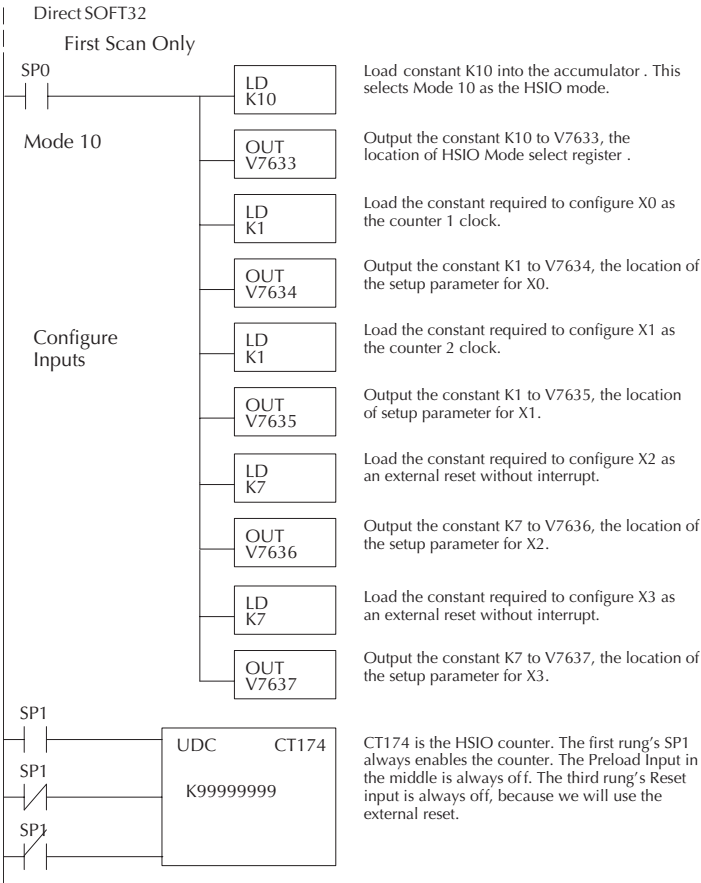
### Direct SOFT32



When the enable input is energized, the up/down counter CT174 will respond to pulses on X0 and increment. The updown counter CT176 will respond to pulses on X1 and increment. The reset input contact behaves in a logical OR fashion with the physical reset input. X2 (when selected) resets counter 1. X3 (when selected) resets counter 2. So, the high speed counter can receive a reset form either the contact(s) on the reset rung in the ladder, OR the external reset X2 or X3, if you have configured X2 or X3 as an external reset.

## Program Example 1: Counter Without Presets

The following example is the simplest way to use the high-speed counters, which does not use the presets and special relays in the interrupt routine. The program configures the HSIO circuit for Mode 10 operation, so X0 is automatically the counter clock input for the first counter, and X1 is the counter clock input for the second counter. It uses the Compare-double (CMPD) instruction to cause action at certain count values. Note that this allows you to have more than 24 “presets”. Then it configures X2 and X3 to be the external reset of the counter.

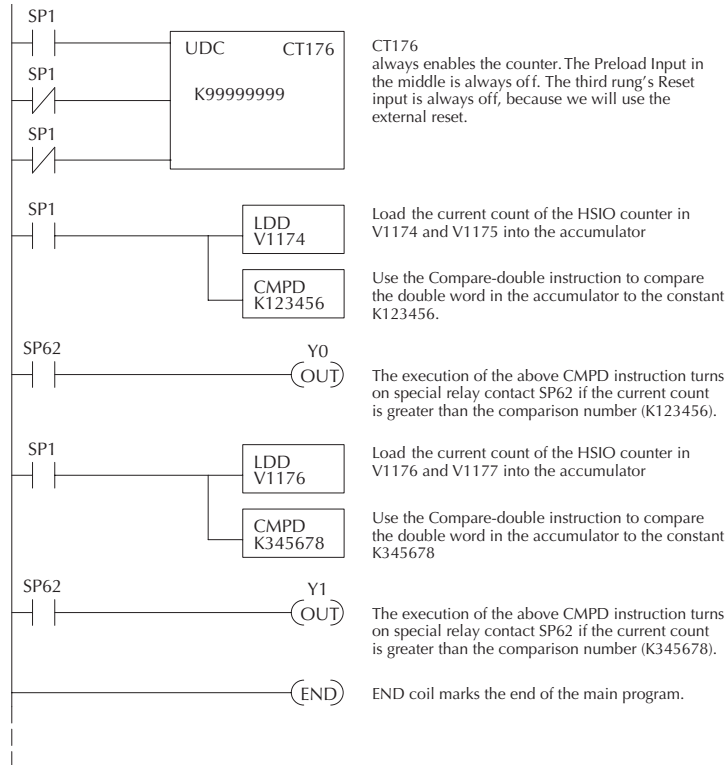


continued on next page

## Program Example Cont'd

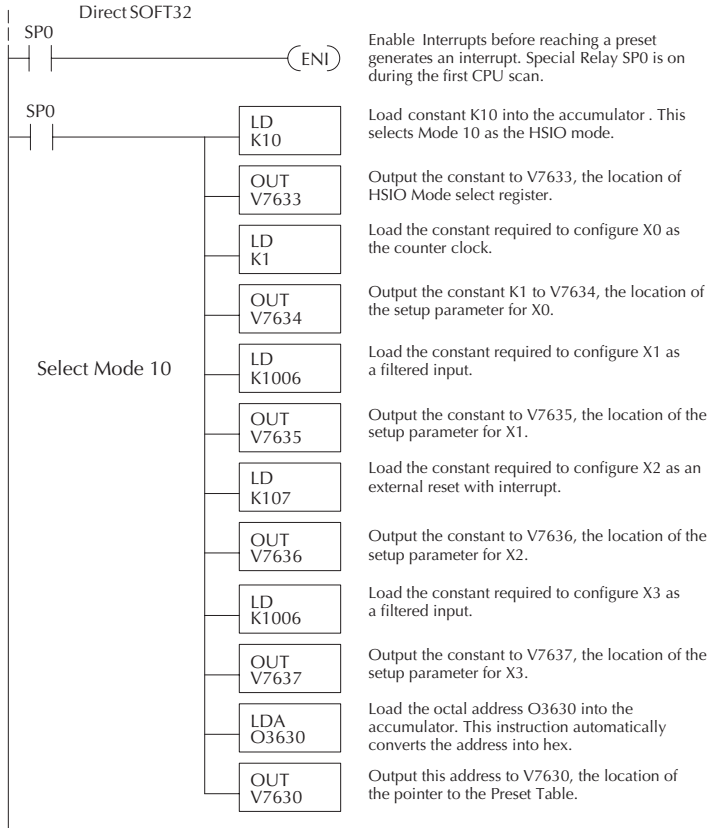
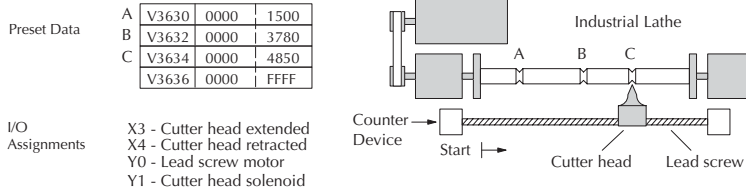
The compare double instructions below use the current count of the HSIO counter to turn on Y0 and Y1. This technique can make more than 24 comparisons, but it is scan-time dependent. However, use the 24 built-in presets with the interrupt routine if your application needs a very fast response time, as shown in the next example.

continued from previous page



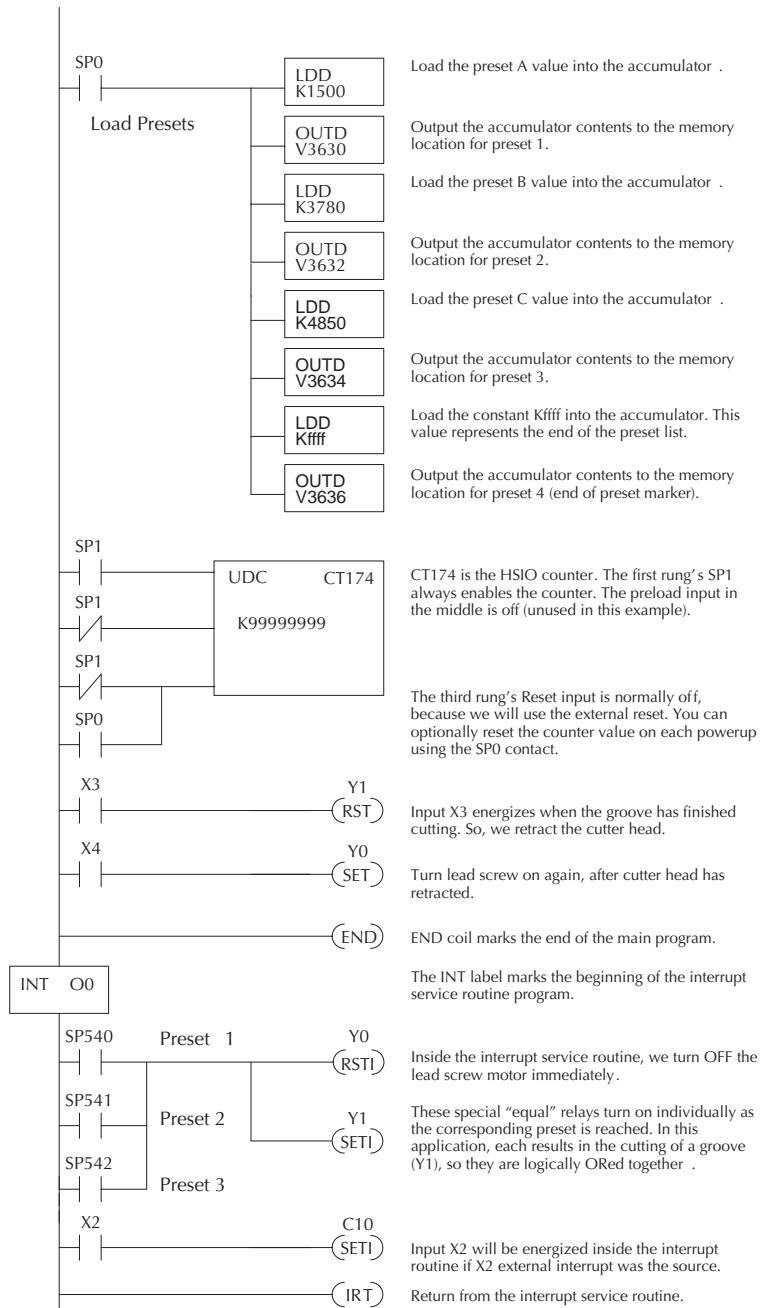
## Program Example 2: Counter With Presets

The following example shows how to program the HSIO circuit to trigger on three preset values. You may recall the industrial lathe example from the beginning of this chapter. This example program shows how to control the lathe cutter head to make three grooves in the work-piece at precise positions. When the lead screw turns, the counter device generates pulses which the DL06 can count. The three preset variables A, B, and C represent the positions (number of pulses) corresponding to each of the three grooves. In this example, only one high-speed counter is used. The second counter could be used in the same manner.



continued on next page

continued from previous page

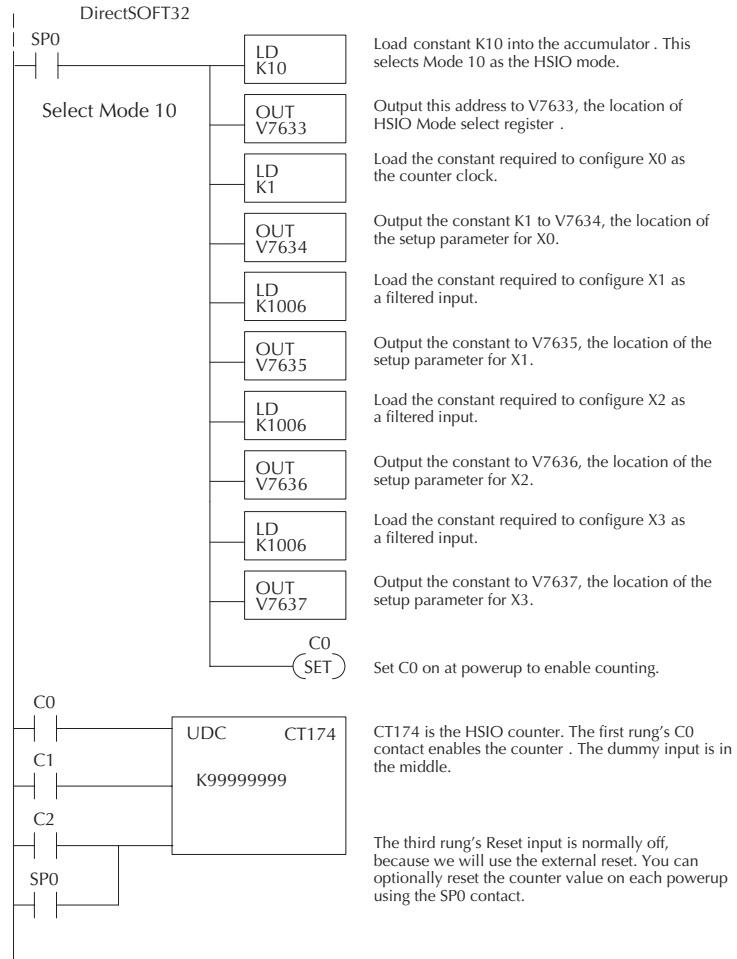




Some applications will require a different type of action at each preset. It is possible for the interrupt routine to distinguish one preset event from another, by turning on a unique output for each equal relay contact SPxxx. We can determine the source of the interrupt by examining the equal relay contacts individually, as well as X2. The X2 contact will be on (inside the interrupt routine only) if the interrupt was caused by the external reset, X2 input.

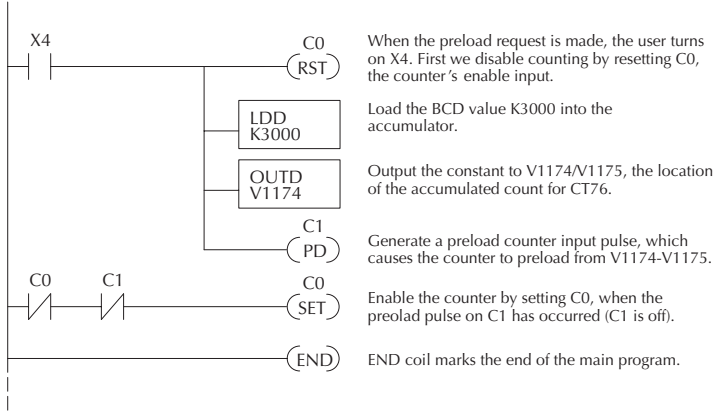
## Program Example 3: Counter With Preload

The following example shows how you can preload the current count with another value. When the preload command input (X4 in this example) is energized, we disable the counter from counting with C0. Then we write the value K3000 to the count register (V1076-V1077). We preload the current count of the counter with K3000. When the preload command (X4) is turned off, the counter resumes counting any pulses, but now starting from K3000. In this example, only one high-speed counter is used. The second counter could be used in the same manner.



continued on next page

continued from last page



### Troubleshooting Guide for Mode 10

If you're having trouble with Mode 10 operation, please study the following symptoms and possible causes. The most common problems are listed below.

#### **Symptom: The counter does not count.**

Possible causes:

1. **Field sensor and wiring** – Verify that the encoder, proximity switch, or counter actually turns on and illuminates the status LED for X0 (counter 1) and X1 (counter 2). The problem could be due to sinking-sourcing wiring problem, etc. Remember to check the signal ground connection. Also verify that the pulse on-time is long enough for the PLC to recognize it.
2. **Configuration** – use the Data View window to check the configuration parameters. V7633 must be set to 10, and V7634 must be set to 1 or 101 to enable the first high-speed counter. V7635 must be set to 1 or 101 to enable the second high-speed counter.
3. **Stuck in reset** – check the input status of the reset input, X2 and X3. If X2 is on, the counter will not count because it is being held in reset.
4. **Ladder program** – make sure you are using counter CT174 and CT176 in your program. The top input is the enable signal for the counter. It must be on before the counter will count. The middle input is the dummy input. The bottom input is the counter reset, and must be off during counting.

#### **Symptom: The counter counts but the presets do not function.**

Possible causes:

1. **Configuration** – Ensure the preset values are correct. The presets are 32-bit BCD values having a range of 0 to 99999999. Make sure you write all 32 bits to the reserved locations by using the LDD and OUTD instructions. Use only even-numbered addresses, from V3630 to V3767. If using less than 24 presets, be sure to place "0000FFFF," "0000FF00," or "000000FF" in the location after the last preset used.
2. **Interrupt routine** – Only use Interrupt #0. Make sure the interrupt has been enabled by executing an ENI instruction prior to needing the interrupt. The interrupt routine must be placed after the main program, using the INT label and ending with an interrupt return IRT.
3. **Special relays** – Check the special relay numbers in your program. Use SP540 for Preset 1, SP541 for Preset 2, etc. Remember that only one special equal relay contact is on at a time. When the counter value reaches the next preset, the SP contact which is on now goes off and the next one turns on.

#### **Symptom: The counter counts up but will not reset.**

Possible causes:

1. Check the LED status indicator for X2 (counter 1) and X3 (counter 2) to make sure it is active when you want a reset. Or, if you are using an internal reset, use the status mode of DirectSOFT32 to monitor the reset input to the counter.

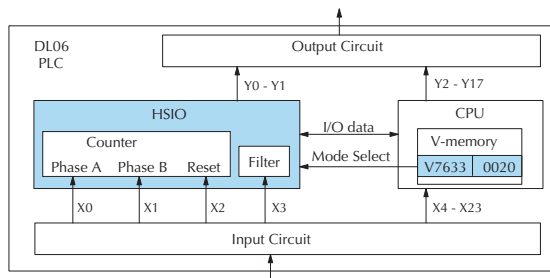
## Mode 20: Up/Down Counter

### Purpose

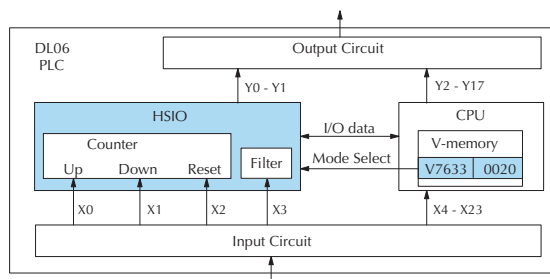
The counter in the HSIO circuit can count up/down signals from two separate sources (i.e. two single channel encoders) or two quadrature signal pulses. Quadrature signals are commonly generated from incremental encoders, which may be rotary or linear. The up/down counter has a range from -8388608 to 8388607. Using CT174 and CT175, the quadrature counter can count up to a 7 kHz rate.

### Functional Block Diagram

The diagram below shows HSIO functionality in Mode 20. When the lower byte of HSIO Mode register V7633 contains a BCD “20”, the up/down counter in the HSIO circuit is enabled. For quadrature counting, input X0 is dedicated to the Phase A quadrature signal, and input X1 receives Phase B signal. X2 is dedicated to reset the counter to zero value when energized.



For standard up/down counting, input X0 is dedicated to the up counting signal, and input X1 is dedicated to the down counting signal. The X2 input resets the counter to zero when energized.

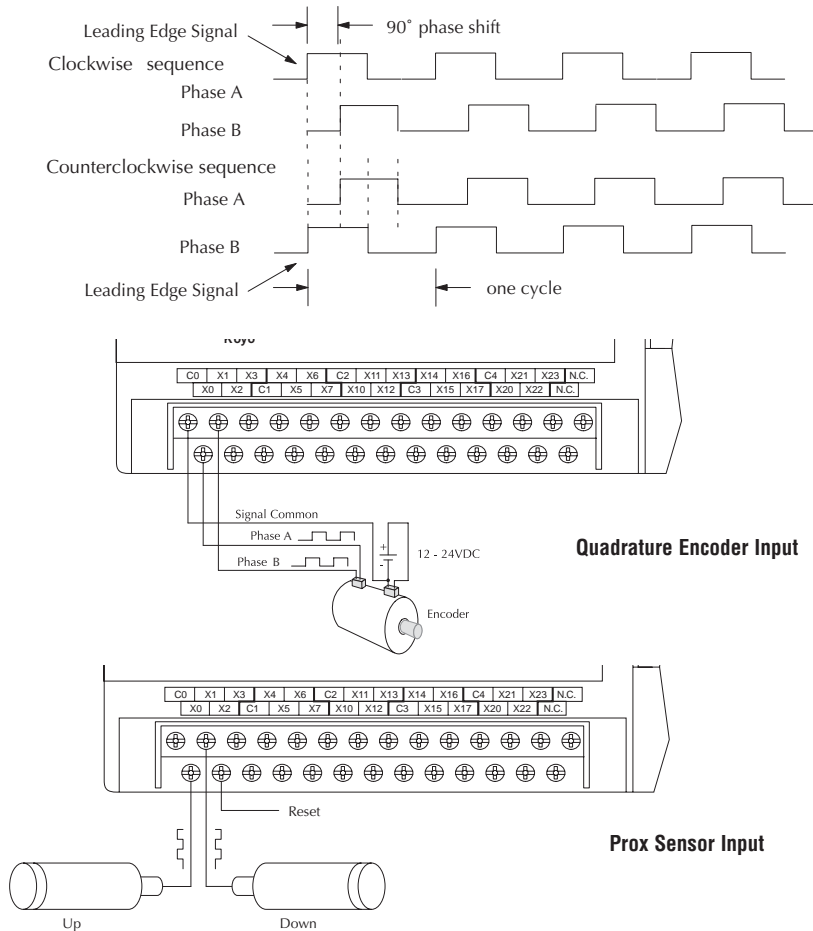


## Quadrature Encoder Signals

Quadrature encoder signals contain position and direction information, while their frequency represents speed of motion. Phase A and B signals shown below are phase-shifted 90 degrees, thus the quadrature name. When the rising edge of Phase A precedes Phase B's leading edge (indicates clockwise motion by convention), the HSIO counter counts UP. If Phase B's rising edge precedes Phase A's rising edge (indicates counter-clockwise motion), the counter counts DOWN.

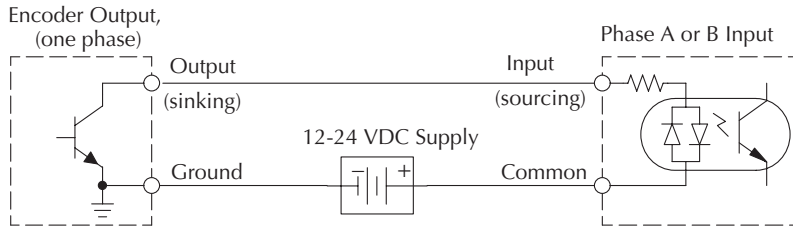
## Wiring Diagram

A general wiring diagram for encoders to the DL06 in HSIO Mode 20 is shown below. Encoders with sinking outputs (NPN open collector) are probably the best choice for interfacing. If the encoder sources to the inputs, it must output 12 to 24 VDC. Note that encoders with 5V sourcing outputs will not work with DL06 inputs.

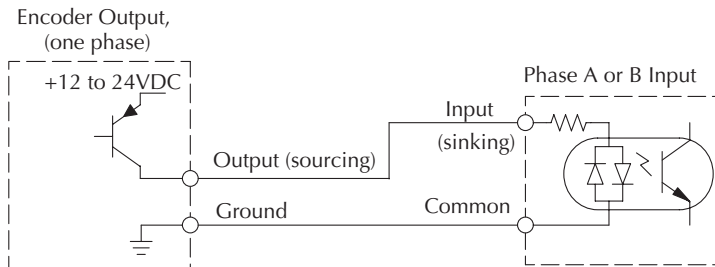


### Interfacing to Encoder Outputs

The DL06's DC inputs are flexible in that they detect current flow in either direction, so they can be wired to an encoder with either sourcing or sinking outputs. In the following circuit, an encoder has open-collector NPN transistor outputs. It sinks current from the PLC input point, which sources current from the PLC input point, which sources current. The power supply can be the +24VDC auxiliary supply or another supply (+12VDC or +24VDC), as long as the input specifications are met.

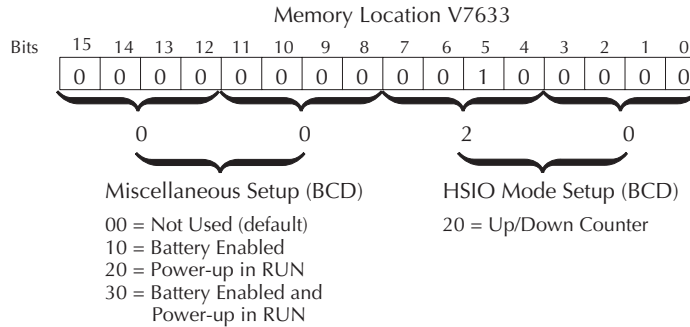


In the next circuit, an encoder has open-emitter PNP transistor outputs. It sources current to the PLC input point, which sinks the current back to ground. Since the encoder sources current, no additional power supply is required. However, note that the encoder output must be 12 to 24 volts (5V encoder outputs will not work).



## Setup for Mode 20

Recall that V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 20 in the lower byte of V7633 to select the High-Speed Counter Mode.



Choose the most convenient method of programming V7633 from the following:

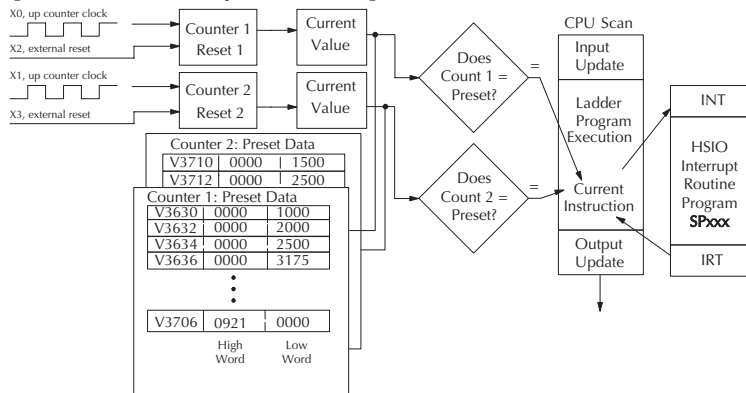
- Include load and out instructions in your ladder program
- *DirectSOFT*'s memory editor
- Use the Handheld Programmer D2-HPP

We recommend using the first method above so that the HSIO setup becomes an integral part of your application program. An example program later in this section shows how to do this.

## Presets and Special Relays

The goal of counting is to cause a particular action to occur when the count reaches a preset value. Refer to the figure below. Each counter features 24 presets, which you can program. A preset is a number you select and store so that the counter will continuously compare the current count with the preset. When the two are equal, a special relay contact is energized and program execution jumps to the interrupt routine.

We recommend using the special relay(s) in the interrupt service routine to cause any immediate action you desire. After the interrupt service routine is complete, the CPU returns to the ladder program, resuming program execution from the point of interruption. The compare function is ready for the next preset event.





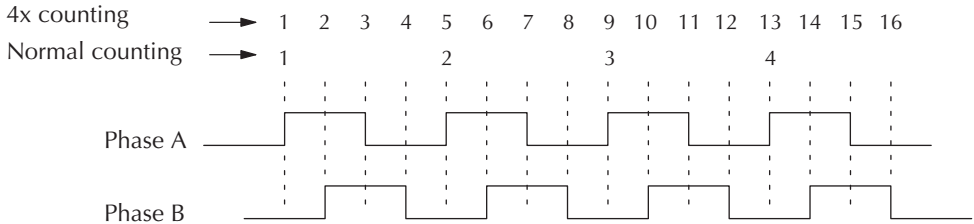
## X Input Configuration

The configurable discrete input options for High-Speed Counter Mode are listed in the table below. The section on Mode 60 operation at the end of this chapter describes programming the filter time constants.

### Mode 20 Up/Down Counter

Input	Configuration Register	Function	Hex Code Required
<b>X0</b>	V7634	Up counting	0202 (standard, absolute)
			0302 (standard, incremental)
		Phase A	0002 (quadrature, absolute) (default)
			0102 (quadrature, incremental)
			1002 (quadrature, absolute) 4x counting*
			1102 (quadrature, incremental) 4x counting*
<b>X1</b>	V7635	Down counting or Phase B	0000
<b>X2</b>	V7636	Counter Reset (no interrupt)	0007** (default) 0207**
		Counter Reset (with interrupt)	0107** 0307**
		Pulse input	0005
		Filtered input	xx06 (xx = filter time, 0 - 99ms (BCD))
<b>X3</b>	V7637	Pulse input	0005
		Filtered input	xx06 (xx = filter time, 0 - 99ms (BCD)) (default)

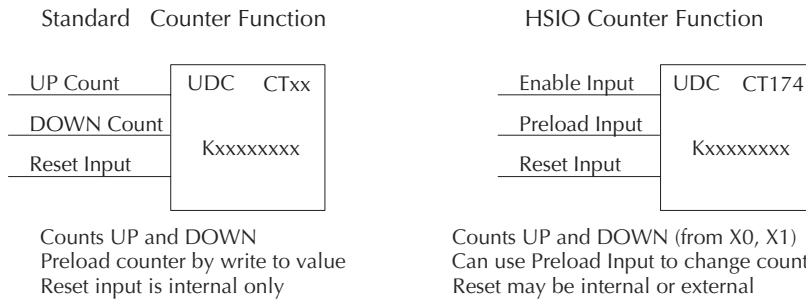
\* With this feature, you can count 4 times more with the same encoder.



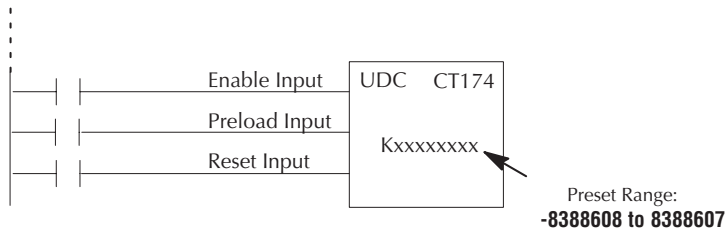
\*\* With the counter reset you have the option of a normal reset or a faster reset. However, the fast reset does not recognize changed preset values during program execution. When '0007' or '0107' are set in V7636 and preset values are changed during program execution, the DL06 recognizes the changed preset values at the time of the reset. When '0207' or '0307' are set in V7636 the CPU does not check for changed preset values, so the DL06 has a faster reset time.

## Writing Your Control Program

The mnemonic for the counter is UDC (up-down counter). The DL06 can have up to 128 counters, labeled CT0 through CT177. The quadrature counter in the HSIO circuit is accessed in ladder logic by using UDC CT174. It uses counter registers CT174 and CT175 exclusively when the HSIO mode 20 is active (otherwise, CT174 and CT175 are available for standard counter use). The HSIO counter needs two registers because it is a double-word counter. It has three inputs as shown. The first input is the enable signal, the middle is a preload (write), and the bottom is the reset. The enable input must be on before the counter will count. The enable input must be off during a preload.



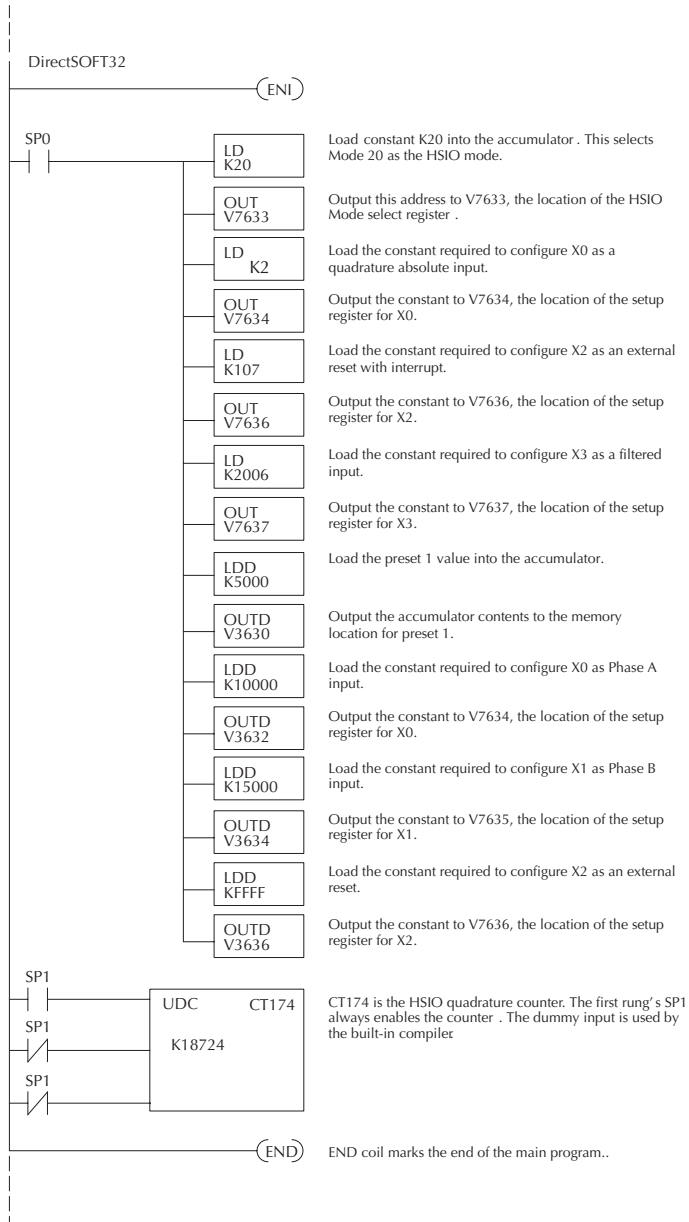
The next figure shows the how the HSIO quadrature counter will appear in a ladder program.



When the enable input is energized, the counter will respond to quadrature pulses on X0 and X1, incrementing or decrementing the counter at CT174 – CT175. The reset input contact behaves in a logical OR fashion with the physical reset input X2. This means the quadrature counter can receive a reset from either the contact(s) on the reset rung in the ladder, OR the external reset X2.

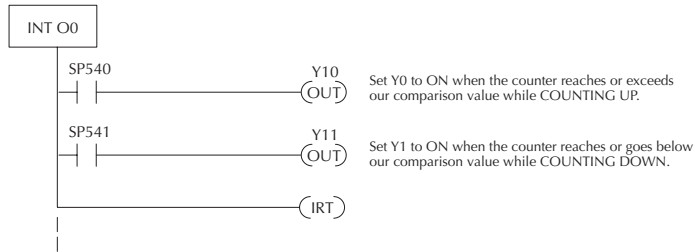
### Program Example 1: Quadrature Counting with an Interrupt

Below is a simple example of how quadrature counting with an interrupt can be programmed.



continued on next page

continued from last page



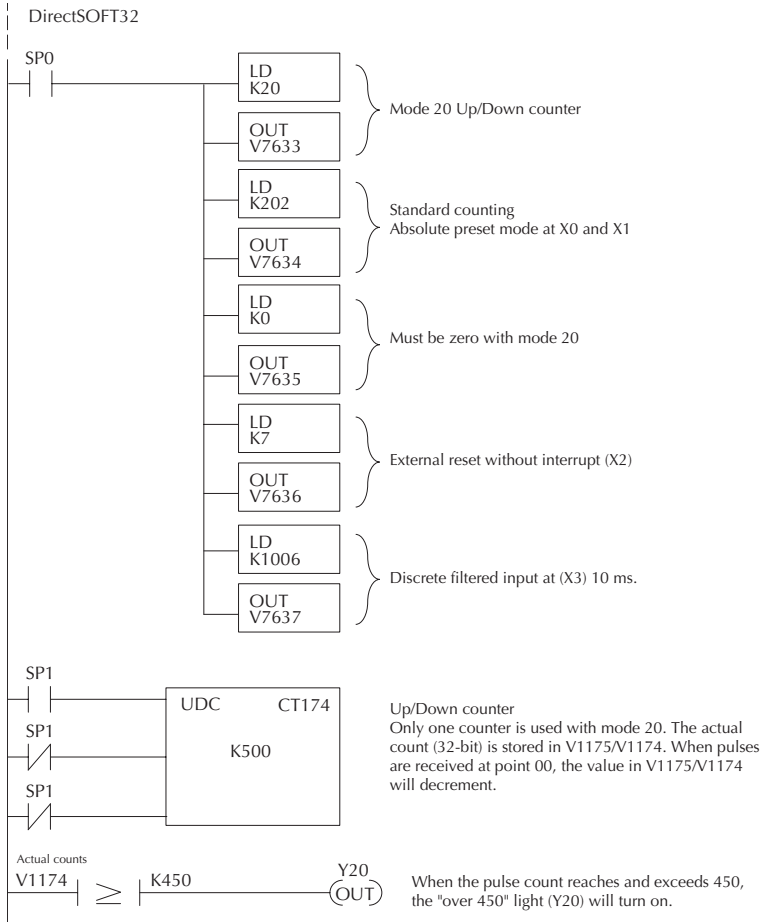
The Load Accumulator instructions have set up the V-memory as required, i.e. 20 in V7633 for the mode and 0202 in V7634 to designate the standard up/down with the absolute preset mode. By placing 0107 in V7636, an external reset for counter CT174 is selected and it will execute interrupt 0 on the rising edge of the reset. Presets for up/down counting have been stored in memory locations V3630 through V3635. The next even numbered location following this has FFFF to indicate we have no more presets.

### Program Example 2: Up/Down Counting with Standard Inputs

In this example, there is a conveyor belt “A” that transports bottles to be inspected. During the course of the process, one sensor is keeping track of the bottles that are going onto belt “A” for inspection, and another sensor is keeping track of how many bottles are being removed to the finished product line.

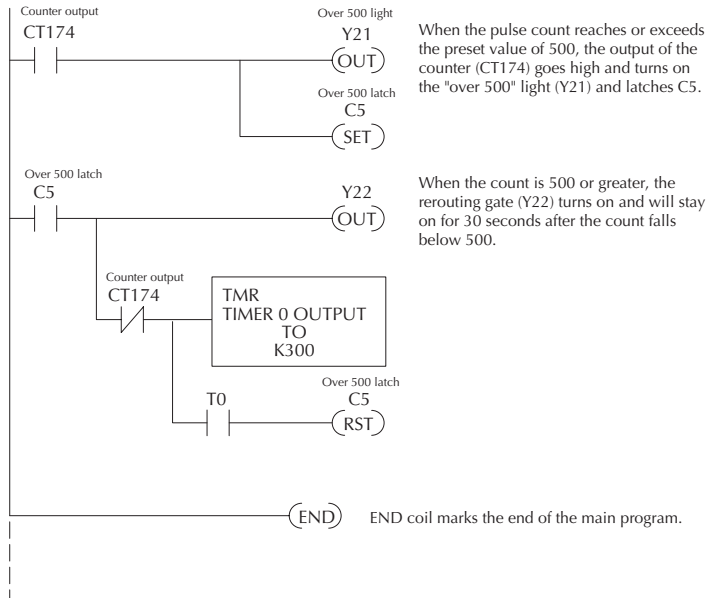
When we have reached 500 bottles in the process, an “over 500” light turns on and a rerouting gate is activated to channel the incoming bottles to conveyor belt “B”. The rerouting gate will stay activated for 30 seconds after the conveyor belt “A” contains less than 500 bottles.

The program below shows how ladder logic might be written to handle the job. Note the use of V1174. This memory location stores the current count for CT174 which is used with the DL06.



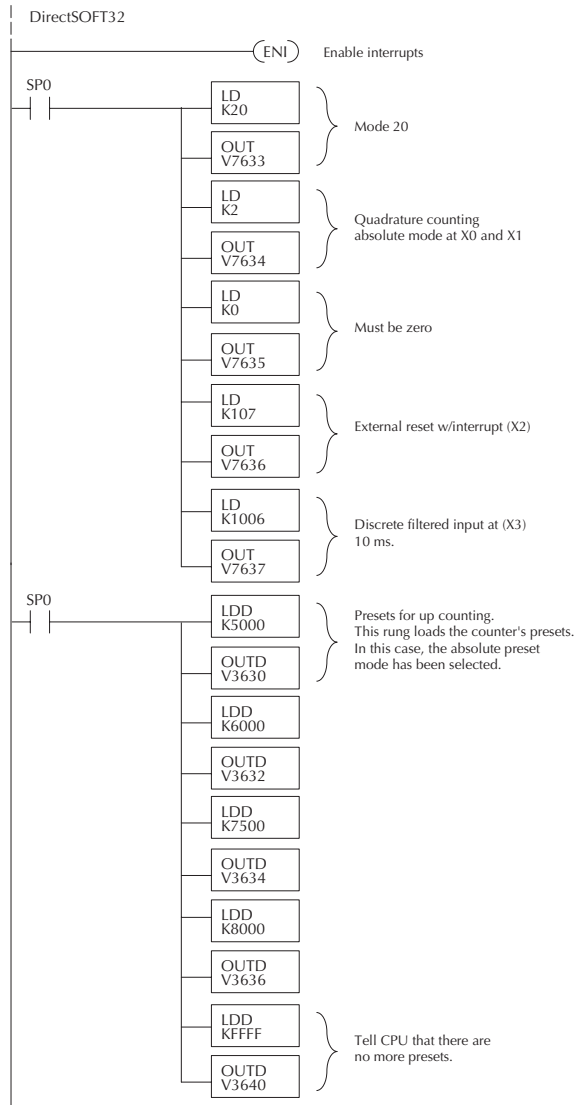
Continued on next page.

continued from previous page



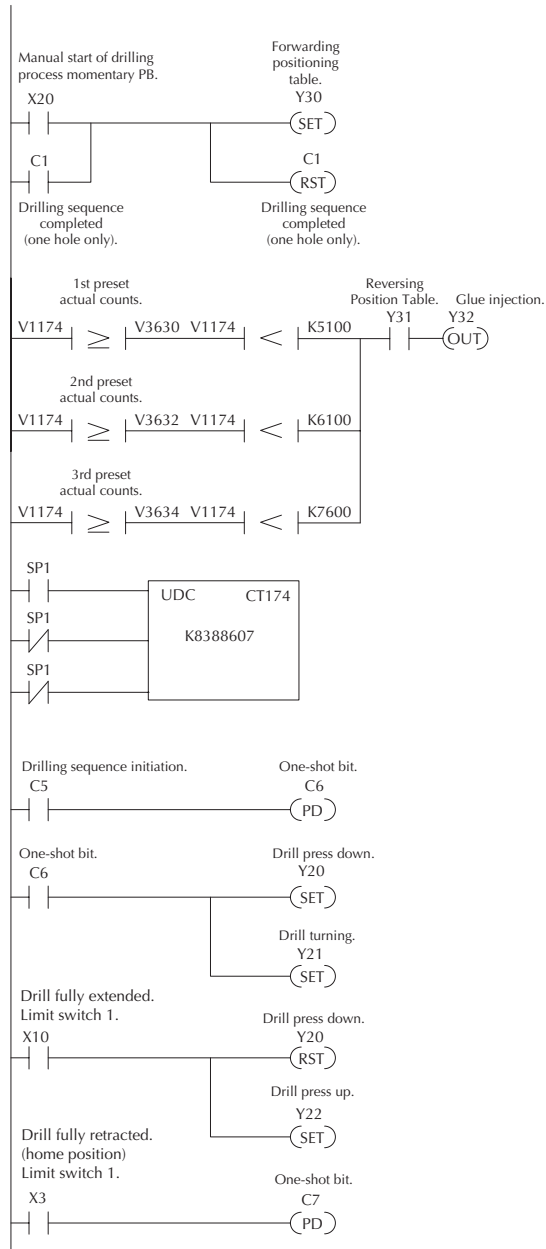
### Program Example 3: Quadrature Counting

In this example, a wooden workpiece is being drilled with 3 holes and then the holes are injected with glue for dowels to be inserted at another workstation. A quadrature encoder is connected to a positioning table which is moving a drill press horizontally over the workpiece. The positioning table will stop and the drill press will lower to drill a hole in an exact location. After the three holes are drilled in the workpiece, the positioning table reverses direction and injects glue into the same holes.



Continued on next page.

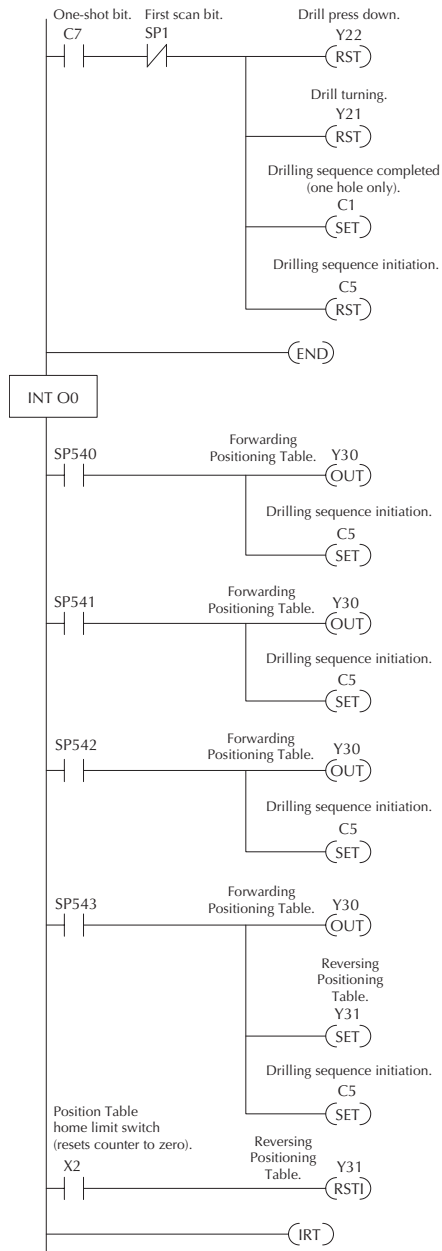
Continued from previous page.



Continued on next page.



Continued from previous page.



### Troubleshooting Guide for Mode 20

If you're having trouble with Mode 20 operation, please study the following symptoms and possible causes. The most common problems are listed below.

#### **Symptom: The counter does not count.**

Possible causes:

1. **Field sensor and wiring** – Verify that the encoder or other field device inputs actually turn on and illuminates the status LEDs for X0 and X1. A standard incremental encoder will visibly, alternately turn on the LEDs for X0 and X1 when rotating slowly (1 RPM). Or, the problem could be due to a sinking-sourcing wiring problem, etc. Remember to check the signal ground connection. Also verify that the pulse on-time, duty cycle, voltage level, and frequency are within the input specifications.
2. **Configuration** – make sure all of the configuration parameters are correct. V7633 must be set to 20, and V7634 must be set to “0002” to enable the Phase A input, and V7635 must be set to “0000” to enable the Phase B input.
3. **Stuck in reset** – check the input status of the reset input, X2. If X2 is on, the counter will not count because it is being held in reset.
4. **Ladder program** – make sure you are using counter CT174 in your program. The top input is the enable signal for the counter. It must be on before the counter will count. The middle input is the dummy input and must be off for the counter to count. The bottom input is the counter reset, and must be off during counting.

#### **Symptom: The counter counts in the wrong direction (up instead of down, and visa-versa).**

Possible causes:

1. **Channel A and B assignment** – It's possible that Channel A and B assignments of the encoder wires is backwards from the desired rotation/counting orientation. Just swap the X0 and X1 inputs, and the counting direction will be reversed.

#### **Symptom: The counter counts up and down but will not reset.**

Possible causes:

1. Check the LED status indicator for X2 to make sure it is active when you want a reset. Also verify the configuration register V7636 for X2 is set to 7. Or, if you are using an internal reset, use the status mode of DirectSOFT32 to monitor the reset input to the counter.

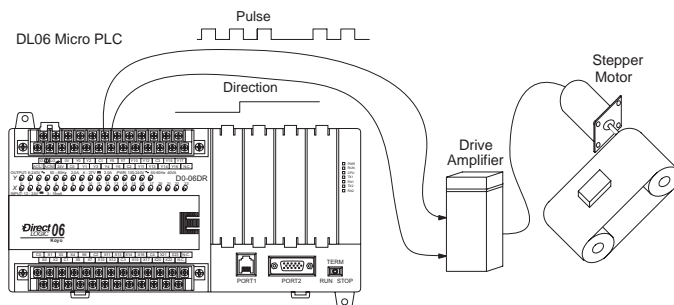
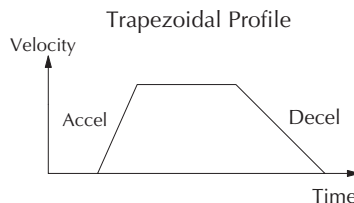
# Mode 30: Pulse Output

## Purpose

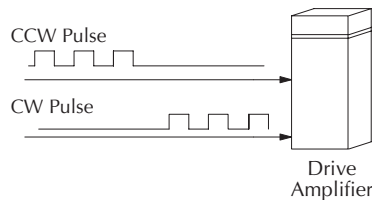
The HSIO circuit in Mode 30 generates output pulse trains suitable for open-loop control of a single-axis motion positioning system. It generates pulse (stepper increment) and direction signals which you can connect to motor drive systems and perform various types of motion control. Using Mode 30 Pulse Output, you can select from three profile types detailed later in this chapter:

- Automatic Trapezoidal – Accel Slope to Target Velocity to Decel Slope
- Step Trapezoidal – User defined step acceleration/deceleration and target velocity
- Velocity Control – Speed and Direction only

The HSIO circuit becomes a high-speed pulse generator (up to 10 kHz) in Mode 30. By programming acceleration and deceleration values, position and velocity target values, the HSIO function automatically calculates the entire motion profile. The figure below shows the DL06 generating pulse and direction signals to the drive amplifier of a stepper positioning system. The pulses accomplish the profile independently and without interruption to ladder program execution in the CPU.



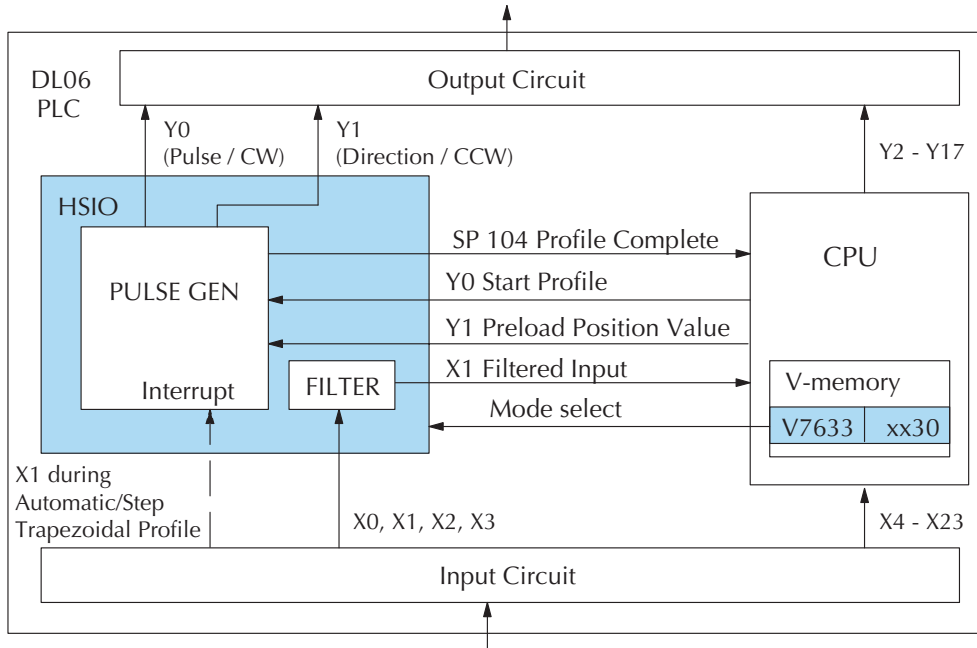
In the figure above, the DL06 generates pulse and direction signals. Each pulse represents the smallest increment of motion to the positioning system (such as one step or micro-step to a stepper system). Alternatively, the HSIO Pulse Output Mode may be configured to deliver counter clock-wise (CCW) and clock-wise (CW) pulse signals as shown to the right.



**NOTE:** The pulse output is designed for open loop stepper motor systems. This, plus its minimum velocity of 40 pps make it unsuitable for servo motor control.

## Functional Block Diagram

The diagram below shows HSIO functionality in Mode 30. When the lower byte of HSIO Mode register V7633 contains a BCD “30”, the pulse output capability in the HSIO circuit is enabled. The pulse outputs use Y0 and Y1 terminals on the output connector. Remember that the outputs can only be DC type to operate.



**IMPORTANT NOTE:** In Pulse Output Mode, Y0 and Y1 references are redefined or are used differently in two ways. Physical references refer to terminal screws, while logical references refer to I/O references in the ladder program. Please read the items below to understand this very crucial point.

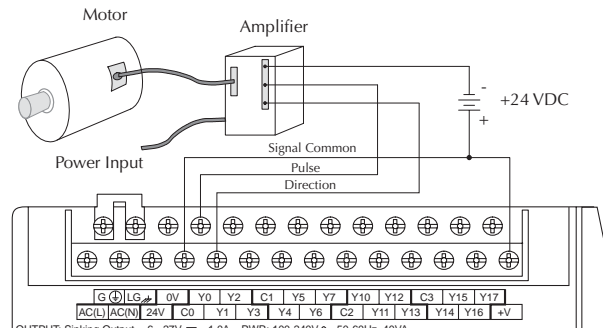
Notice the I/O point assignment and usage in the above diagram:

- X0, X1, X2 and X3 can be filtered inputs or pulse inputs in Pulse Output Mode, and they are available as input contacts to the ladder program.
- X1 behaves as an external interrupt to the pulse generator for automatic/step trapezoidal profiles. In other profile modes, it can be used as a filtered input or pulse input just like X0 (registration mode configuration shown above).
- References “Y0” and “Y1” are used in two different ways. At the discrete output connector, Y0 and Y1 terminals deliver the pulses to the motion system. The ladder program uses logical references Y0 and Y1 to initiate “Start Profile” and “Load Position Value” HSIO functions in Mode 30.

Hopefully, the above discussion will explain why some I/O reference names have dual meanings in Pulse Output Mode. Please read the remainder of this section with care, to avoid confusion about which actual I/O function is being discussed.

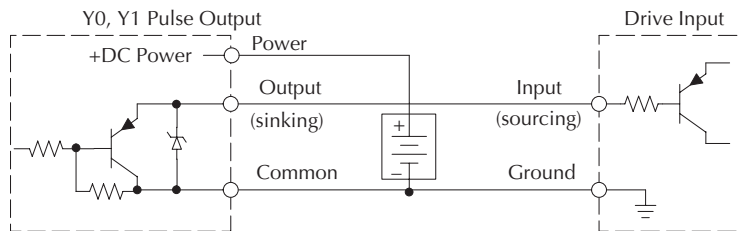
### Wiring Diagram

The generalized wiring diagram below shows pulse outputs Y0 and Y1 connected to the drive amplifier inputs of a motion control system.

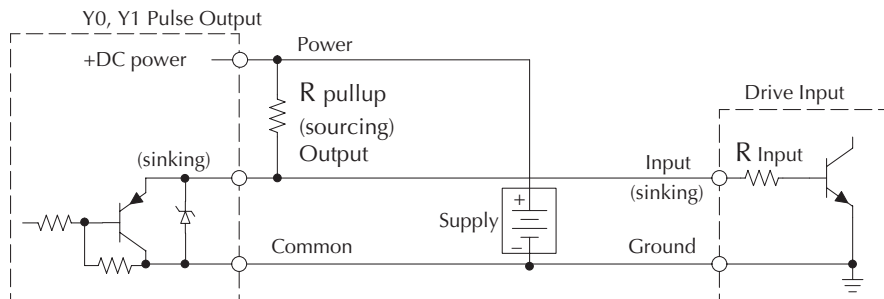


### Interfacing to Drive Inputs

The pulse signals from Y0 and Y1 outputs will typically go to drive input circuits as shown above. It will be helpful to locate equivalent circuit schematics of the drive amplifier. The following diagram shows how to interface to a sourcing drive input circuit.



The following circuit shows how to interface to a sinking drive input using a pullup resistor. Please refer to Chapter 2 to learn how to calculate and install R pullup.



## Motion Profile Specifications

The motion control profiles generated in Pulse Output Mode have the following specifications:

Motion Control Profile Specifications	
Parameter	Specification
Profiles	Automatic Trapezoidal – Accel Slope / Target Velocity / Decel Slope
	Step Trapezoidal - Step Acceleration / Deceleration
	Velocity Control – Speed and Direction only
Position Range	–8388608 to 8388607
Positioning	Absolute / relative command
Velocity Range	40 Hz to 10 kHz
V-memory registers	V3630 to V3652 (Profile Parameter Table)
Current Position	CT174 and CT175 (V1174 and V1175)

## Physical I/O Configuration

The configurable discrete I/O options for Pulse Output Mode are listed in the table below. The CPU uses SP 104 contact to sense “profile complete”. V7632 is used to select pulse/direction or CW/CCW modes for the pulse outputs. Input X1 is dedicated as the external interrupt for use in registration mode.

Physical I/O Configuration			
Input	Configuration Register	Function	Hex Code Required
–	V7632	Y0 = Pulse Y1 = Direction	0103
		Y0 = CW Pulse Y1 = CCW Pulse	0003 (default)
X0	V7634	pulse input	0005
		filtered input	xx06, xx = filter time, 0-9 (BCD) (default)
X1	V7635	pulse input	0005
		filtered input	xx06, xx = filter time, 0-99 (BCD) (default)
X2	V7636	pulse input	0005
		filtered input	xx06, xx = filter time, 0-99 (BCD) (default)
X3	V7637	pulse input	0005
		filtered input	xx06, xx = filter time, 0-99 (BCD) (default)

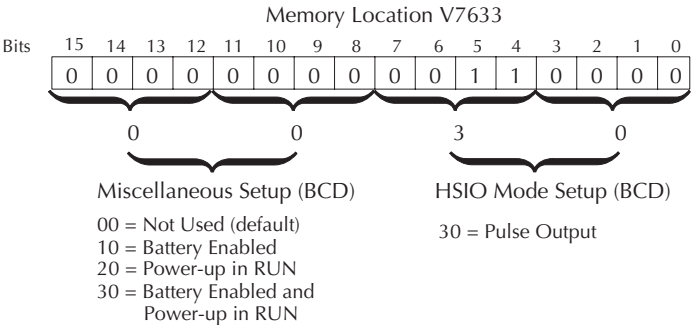
## Logical I/O Functions

The following logical I/O references define functions that allow the HSIO to communicate with the ladder program.

Logical I/O/ Functions	
Logical I/O	Function
SP104	Profile Complete – the HSIO turns on SP104 to the CPU when the profile completes. Goes back off when Start Profile (Y0) turns on.
X1	External Interrupt - If the interrupt feature is selected for the Automatic Trapezoidal profile or the Step Trapezoidal Profile, the DL06 keeps outputting pulses until X1 turns on. After it is on the unit outputs the pulses that are defined as the Target
Y0	Start Profile – the ladder program turns on Y0 to start motion. If turned off before the move completes, motion stops. Turning it on again will start another profile, unless the current position equals the target position.
Y1	Preload Position Value – if motion is stopped and Start Profile is off, you can load a new value in CT174/CT175, and turn on Y1. At that transition, the value in CT174/CT175 becomes the current position.

## Setup for Mode 30

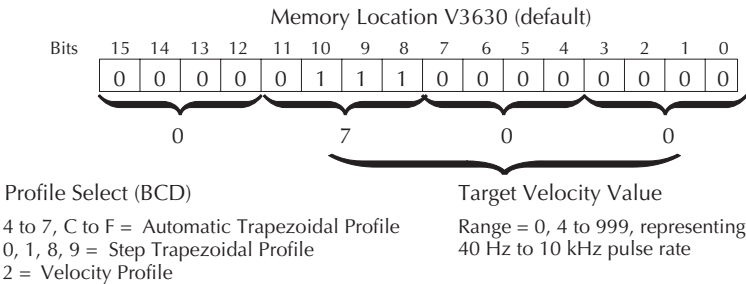
Recall that V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 30 in the lower byte of V7633 to select the High-Speed Counter Mode.



Choose the most convenient method of programming V7633 from the following:

- Include load and out instructions in your ladder program
- *Direct*SOFT32's memory editor
- Use the Handheld Programmer D2-HPP

We recommend using the first method above so that the HSIO setup becomes an integral part of your application program. An example program later in this section shows how to do this.



## Profile / Velocity Select Register

The first location in the Profile Parameter Table stores two key pieces of information. The upper four bits (12–15) select the type of profile required. The lower 12 bits (0–11) select the Target Velocity.

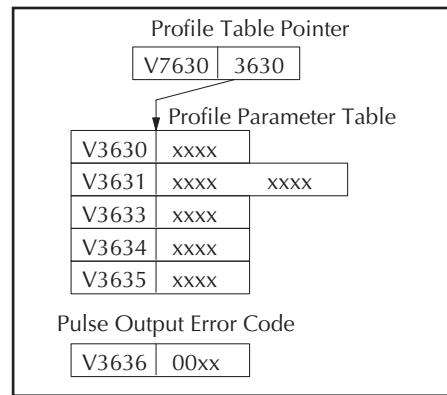
The ladder program must program this location before initiating any of the three profiles. The LD and OUT instruction will write all 16 bits, so be sure to fully specify the full four-digit BCD value for the Profile / Velocity Select Register each time.

The absolute and relative selection determines how the HSIO circuit will interpret your specified target position. Absolute position targets are referenced to zero. Relative position targets are referenced to the current position (previous target position). You may choose whichever reference method is most convenient for your application.

## Profile Parameter Table

V7630 is a pointer location which points to the beginning of the Profile Parameter Table. The default starting location for the profile parameter table is V3630. However, you may change this by programming a different value in V7630. Remember to use the LDA (load address) instruction, converting octal into hex.

The HSIO uses the next V-memory register past the bottom of the profile parameter table to indicate profile errors. See the error table at the end of this section for error code definitions.



## Automatic Trapezoidal Profile

V-Memory	Function	Range	Units
V3630, bits 12–15	Automatic Trapezoidal Profile without Ending Velocity (Ending Velocity is fixed to 0.)	4=absolute w/o interrupt* 5=absolute with interrupt* C=relative w/o interrupt D=relative with interrupt*	–
	Automatic Trapezoidal Profile with Ending Velocity (Use V3637 to set up Ending Velocity.)	6=absolute w/o interrupt* 7=absolute with interrupt* E=relative w/o interrupt F=relative with interrupt*	–
V3630, bits 0–11	Target Velocity	4 to 999 or 0 to 1000	x 10 pps
V3631 / V3632	Target Position**	–8388608 to 8388607	Pulses
V3633	Starting Velocity	4 to 100	x 10 pps
V3634	Acceleration Time	1 to 100	x 100 mS
V3635	Deceleration Time	1 to 100	x 100 mS
V3636	Error Code	(see end of section)	–
V3637	Ending Velocity	4 to 100	x 10 pps

\* If you select to use interrupt, the DL06 will not start looking for your target count until the interrupt X1 is on.

\*\*To set a negative number, put 8 in the most significant digit. For example: -8388608 is 88388608 in V3631 and V3632.



### Step Trapezoidal Profile

V-Memory	Function	Range	Units
V3630, bits 12–15	Step Trapezoidal Profile	0=absolute w/o interrupt 7=absolute with interrupt* 8=relative w/o interrupt 9=relative with interrupt*	–
V3630, bits 0–11	Target Velocity	4 to 999 or 0 for 1000	x 10 pps
V3631 / V3632	Target Position**	–8388608 to 8388607	Pulses
V3633	Step 1 Acceleration	4 to 1000	x 10 pps
V3634	Step 1 Distance	1 to 9999	Pulses
V3635	Step 2 Acceleration	4 to 1000	x 10 pps
V3636	Step 2 Distance	1 to 9999	Pulses
V3637	Step 3 Acceleration	4 to 1000	x 10 pps
V3640	Step 3 Distance	1 to 9999	Pulses
V3641	Step 4 Acceleration	4 to 1000	x 10 pps
V3642	Step 4 Distance	1 to 9999	Pulses
V3643	Step 5 Deceleration	4 to 1000	x 10 pps
V3644	Step 5 Distance	1 to 9999	Pulses
V3645	Step 6 Deceleration	4 to 1000	x 10 pps
V3646	Step 6 Distance	1 to 9999	Pulses
V3647	Step 7 Deceleration	4 to 1000	x 10 pps
V3650	Step 7 Distance	1 to 9999	Pulses
V3651	Step 8 Deceleration	4 to 1000	x 10 pps
V3652	Step 8 Distance	1 to 9999	Pulses

\* If you select to use interrupt, the DL06 will not start looking for your target count until the interrupt X1 is on.

\*\*To set a negative number, put 8 in the most significant digit. For example: -8388608 is 88388608 in V3631 and V3632.

### Velocity Control

V-Memory	Function	Range	Units
V3630	Velocity Profile	2000 only	–
V3631 / 3632	Direction Select	0=CW, 80000000=CCW,	Pulses
V3633	Velocity	4 to 1000	x 10 pps
V3636	Error Code	(see end of section)	–

## Choosing the Profile Type

Pulse Output Mode generates three types of motion profiles. Most applications use one type for most moves. However, each move can be different if required.

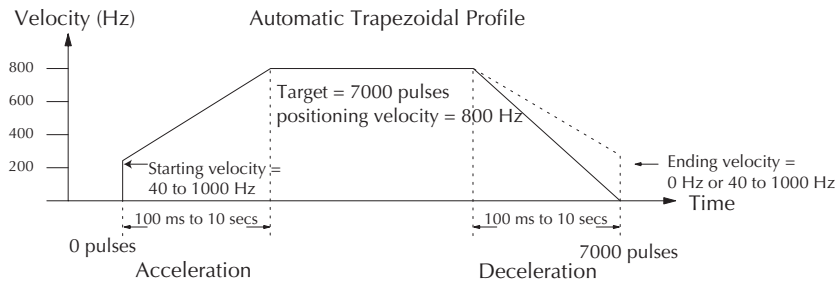
- Automatic Trapezoidal – Accel Slope to Target Velocity to Decel Slope
- Step Trapezoidal – Velocity to Position Control on Interrupt
- Velocity Control – Speed and Direction only

## Automatic Trapezoidal Profile Defined

The automatic trapezoidal profile is the most common positioning profile. It moves the load to a pre-defined target position by creating a move profile. The acceleration slope is applied at the starting position. The deceleration slope is applied backwards from the target position. The remainder of the move in the middle is spent traveling at a defined velocity.

Registration profiles solve a class of motion control problems. In some applications, product material in work moves past a work tool such as a drill station. Shown to the right, registration marks on the scrap area of the work-piece allow a machine tool to register its position relative to the rectangle, to drill properly.

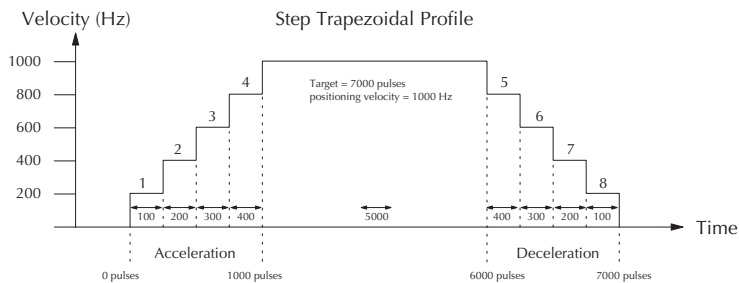
Home search moves allow open-loop motion systems to re-calibrate (preload) the current position value at powerup.



The user determines the starting velocity, the acceleration/deceleration times, and the total number of pulses. The CPU computes the profile from these inputs.

### Step Trapezoidal Profiles Defined

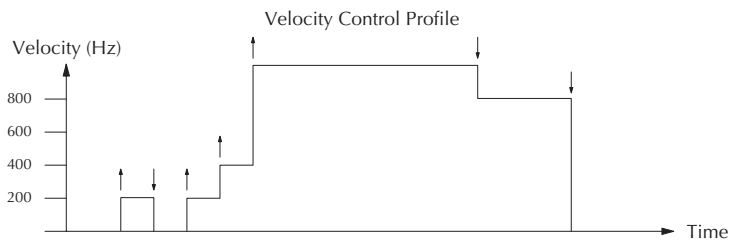
Registration profiles are a combination of velocity and position control modes. The move begins by accelerating to a programmed velocity. The velocity is sustained and the move is of indefinite duration. When an external interrupt signal occurs (due to registration sensing), the profile switches from velocity to position control. The move ends by continuing motion a pre-defined distance past the interrupt point (such as a drill hole location). The deceleration ramp is applied in advance of the target position.



Define steps 1 through 4 for gradual acceleration to the target velocity and define steps 5 through 8 for gradual deceleration from the target velocity. This type of profile is appropriate for applications involving large stepper motors and/or large inertia loads. It can, however, be used to provide gradual ramping in applications involving smaller motors and loads.

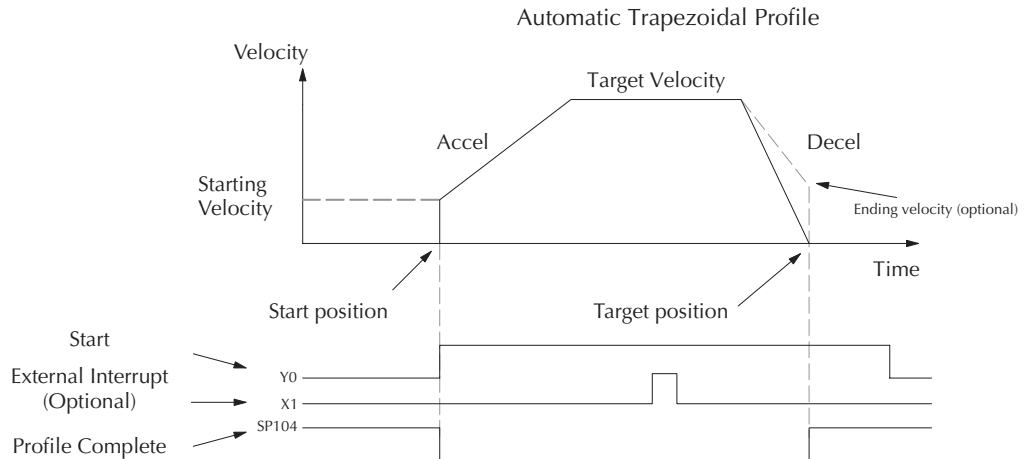
### Velocity Control Defined

The Velocity Control defines only the direction and speed of motion. There is no target position specified, so the move can be of indefinite length. Only the first velocity value needs to be defined. The remaining velocity values can be created while motion is in progress. Arrows in the profile shown indicate velocity changes.



## Automatic Trapezoidal Profile Operation

Starting velocities must be within the range of 40 pps to 1k pps. The remainder of the profile parameters are in the profile parameter table.



The time line of signal traces below the profile indicates the order of events. The HSIO uses logical output Y0 as the Start input to the HSIO, which starts the profile. Immediately the HSIO turns off the Profile Complete signal (SP104), so the ladder program can monitor the progress of the move. Typically a ladder program will monitor this bit so it knows when to initiate the next profile move.

You can also use the external interrupt (X1). Once the external interrupt feature is selected for the profile, the DL06 keeps outputting the pulses until X1 turns on. Then, the DL06 outputs the pulses defined as the target position.

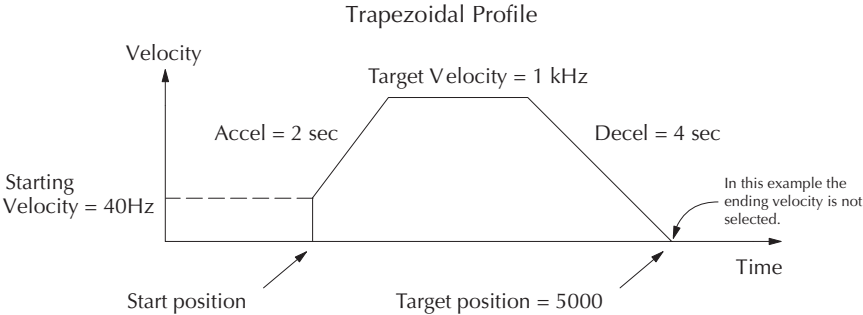
If you are familiar with motion control, you'll notice that we do not have to specify the direction of the move. The HSIO function examines the target position relative to the current position, and automatically outputs the correct direction information to the motor drive.

Notice that the motion accelerates immediately to the starting velocity. This segment is useful in stepper systems so we can jump past low speed areas when low-torque problems or a resonant point in the motor might cause a stall. (When a stepper motor stalls, we have lost the position of the load in open-loop positioning systems). However, it is preferable not to make the starting velocity too large, because the stepper motor will also "slip" some pulses due to the inertia of the system. You can also set up the ending velocity for the same reason.

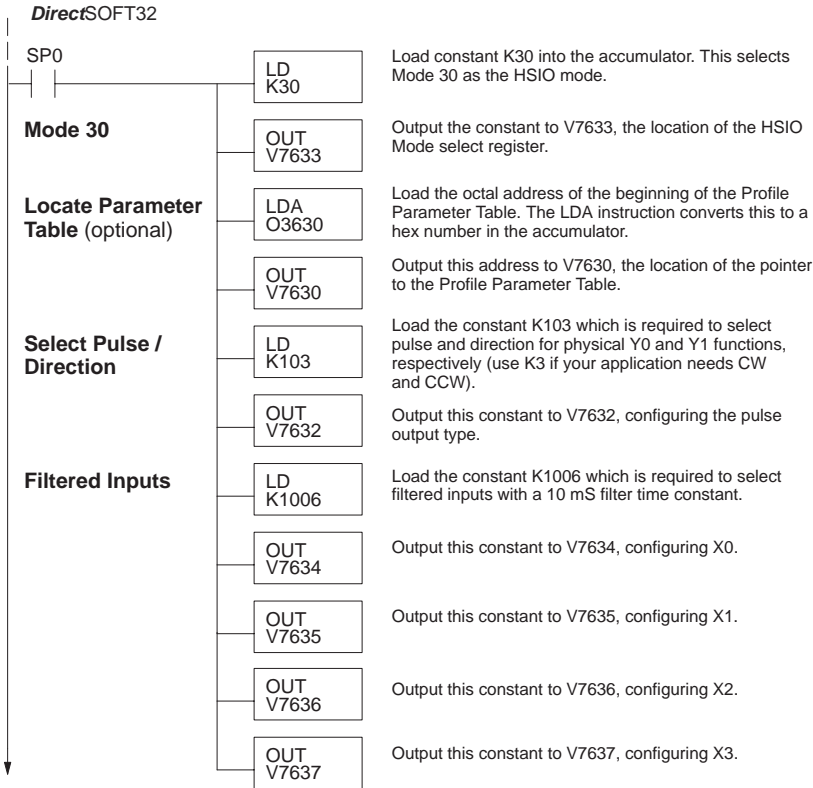
When you need to change the current position value, use logical Y1 output coil to load a new value into the HSIO counter. If the ladder program loads a new value in CT174/CT175 (V1174/V1175), then energizing Y1 will copy that value into the HSIO circuit counter. This must occur before the profile begins, because the HSIO ignores Y1 during motion.

Program Example 1: Automatic Trapezoidal Profile without External Interrupt

The Automatic Trapezoidal Profile we want to perform is drawn and labeled in the following figure. It consists of a non-zero starting velocity, and moderate target velocity.

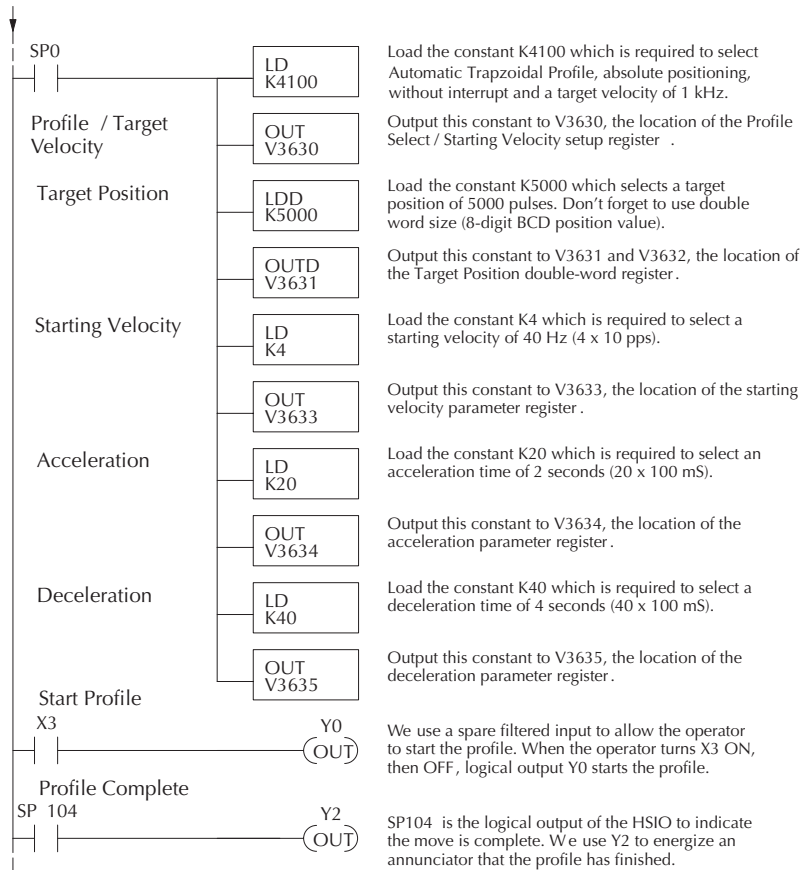


The following program will realize the profile drawn above, when executed. The beginning of the program contains all the necessary setup parameters for Pulse Output Mode 30. We only have to do this once in the program, so we use first-scan contact SP0 to trigger the setup.



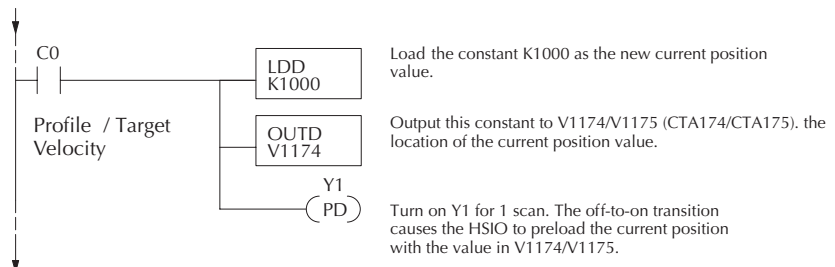
Continued on next page.

Continued from previous page.



## Preload Position Value

At any time you can write (preload) a new position into the current position value. This is often done after a home search (see the registration example programs).



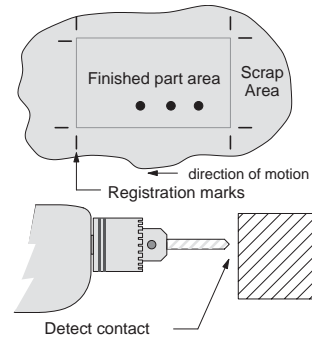
### Program Example 2: Automatic Trapezoidal Profile with External Interrupt

Registration Applications:

1. In a typical application shown to the right, product material in work moves past a work tool such as a drill. Registration marks on the scrap area of the work-piece allow a machine tool to register its position relative to the rectangle, to drill properly.

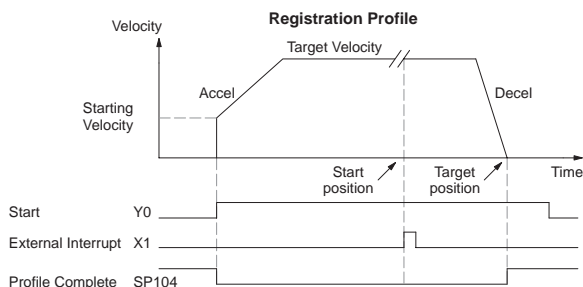
2. In other examples of registration, the work piece is stationary and the tool moves. A drill bit may approach the surface of a part in work, preparing to drill a hole of precise depth.

However, the drill bit length gradually decreases due to tool wear. A method to overcome this is to detect the moment of contact with the part surface on each drill, moving the bit into the part a constant distance after contact. Detect contact



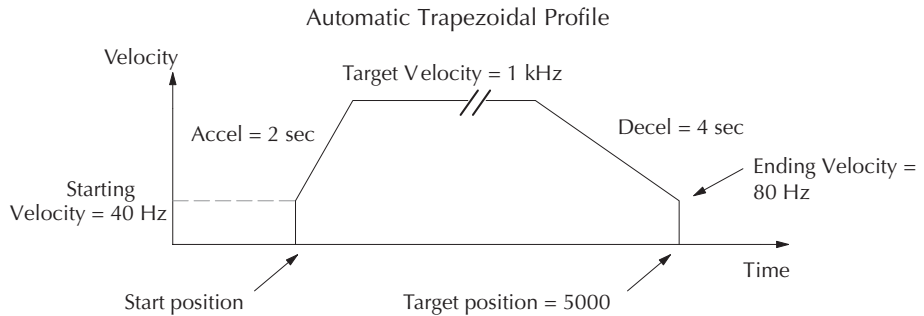
3. The home search move allows a motion system to calibrate its position on startup. In this case, the positioning system makes an indefinite move and waits for the load to pass by a home limit switch. This creates an interrupt at the moment when the load is in a known position. We then stop motion and preload the position value with a number which equates to the physical “home position”.

When an interrupt pulse occurs on physical input X1, the starting position is declared to be the present count (current load position). The velocity control switches to position control, moving the load to the target position. Note that the minimum starting velocity is 40 pps. This instantaneous velocity accommodates stepper motors that can stall at low speeds.

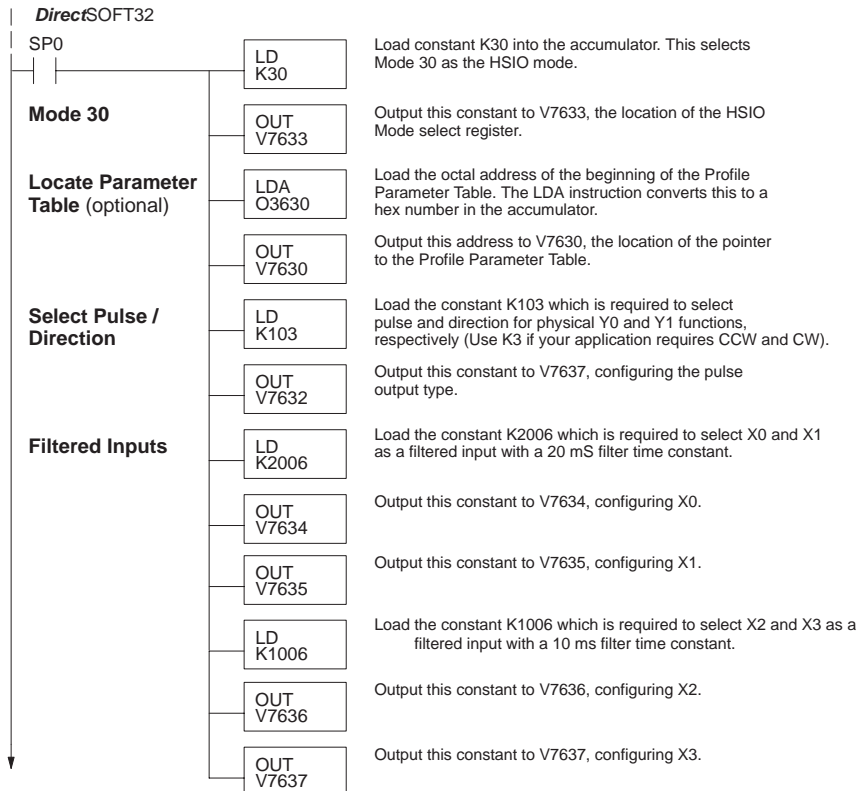


The time line of signal traces below the profile indicates the order of events. The CPU uses logical output Y0 to start the profile. Immediately the HSIO turns off the Profile Complete signal (SP104), so the ladder program can monitor the move's completion by sensing the signal's on state.

The Automatic Trapezoidal profile we want to perform is drawn and labeled in the following figure. It consists of a non-zero starting velocity, and moderate target velocity.

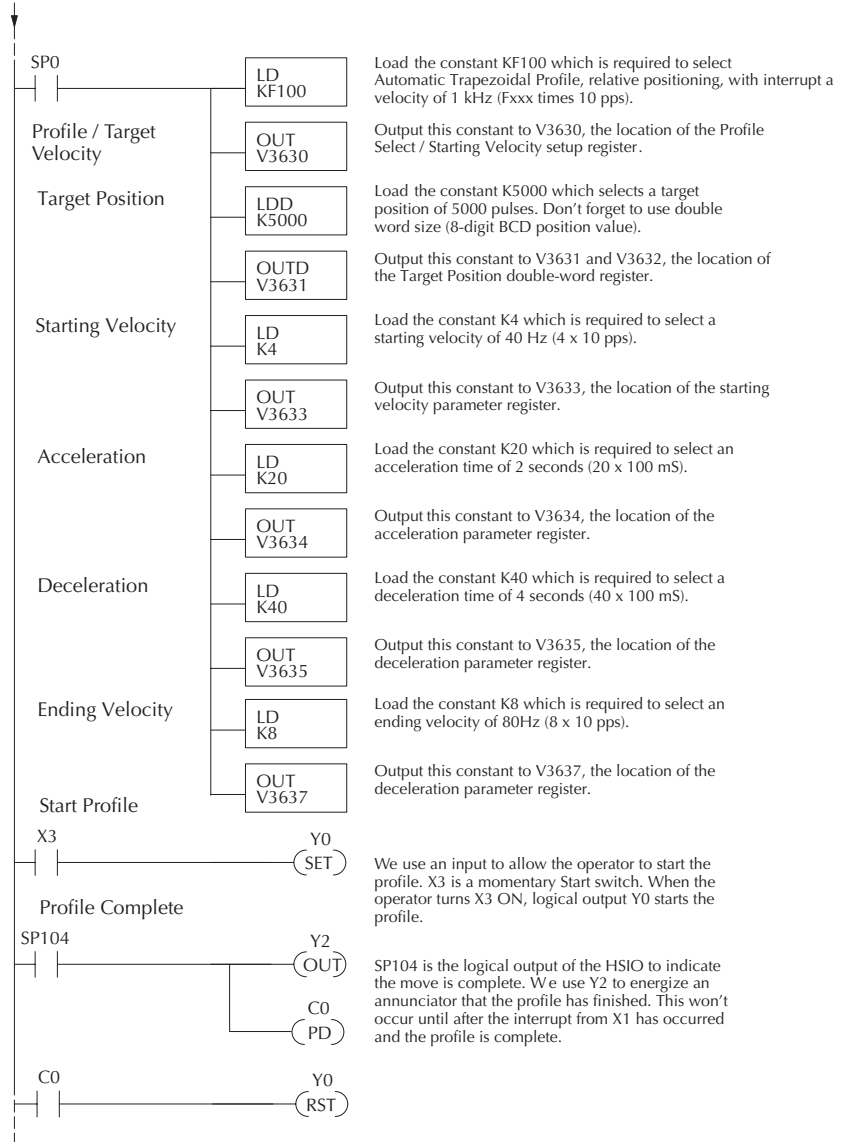


The following program will realize the profile drawn above, when executed. The first program rung contains all the necessary setup parameters. We only have to do this once in the program, so we use first-scan contact SP0 to trigger the setup.





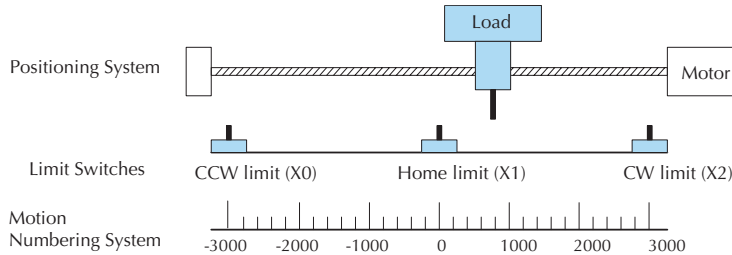
Continued from previous page



The profile will begin when the start input (X3) is given. Then the motion begins an indefinite move, which lasts until an external interrupt on X1 occurs. Then the motion continues on for 5000 more pulses before stopping.

## Program Example 3: Automatic Trapezoidal Profile with Home Search

One of the more challenging aspects of motion control is the establishment of actual position at powerup. This is especially true for open-loop systems which do not have a position feedback device. However, a simple limit switch located at an exact location on the positioning mechanism can provide “position feedback” at one point. For most stepper control systems, this method is a good and economical solution.

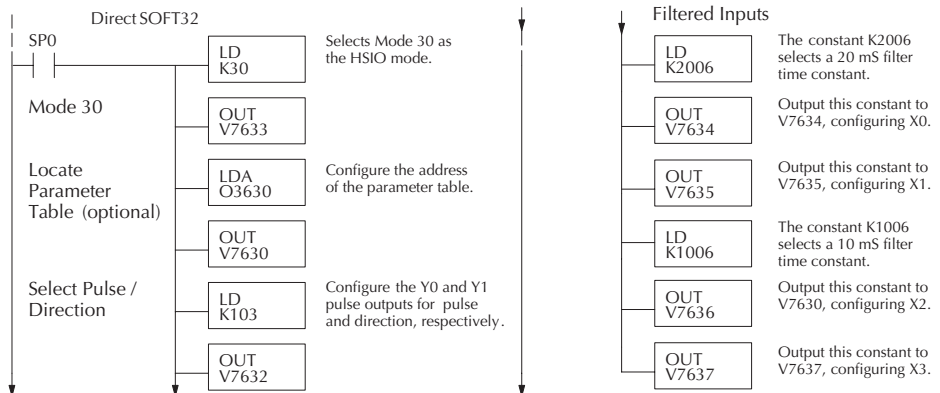


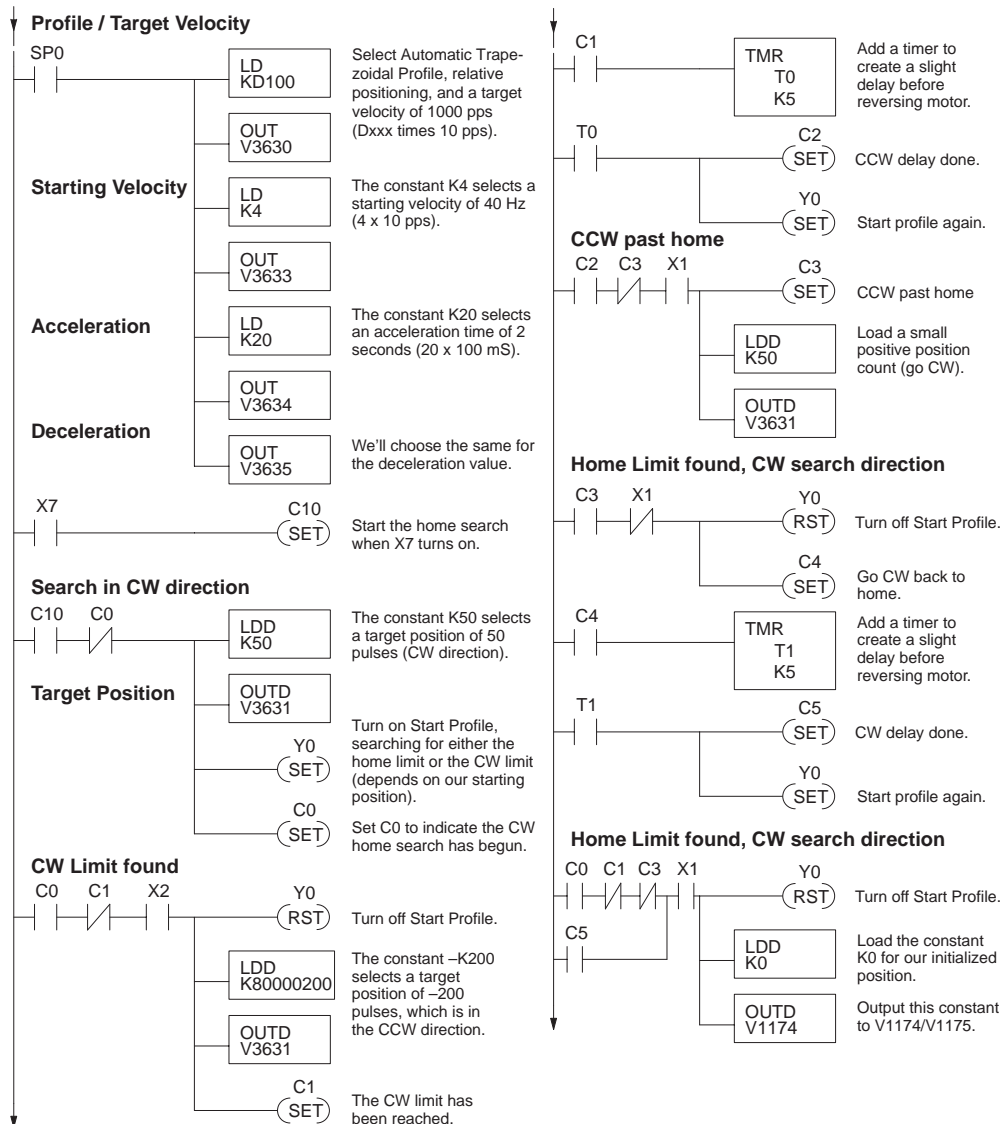
In the drawing above, the load moves left or right depending on the CW/CCW direction of motor rotation. The PLC ladder program senses the CW and CCW limit switches to stop the motor, before the load moves out-of-bounds and damages the machine. The home limit switch is used at powerup to establish the actual position. The numbering system is arbitrary, depending on a machine's engineering units.

At powerup, we do not know whether the load is located to the left or to the right of the home limit switch. Therefore, we will initiate a home search profile, using the registration mode. The home limit switch is wired to X1, causing the interrupt. We choose an arbitrary initial search direction, moving in the CW (left-to-right) direction.

- If the home limit switch closes first, then we stop and initialize the position (this value is typically “0”, but it may be different if preferred).
- However, if the CW limit switch closes first, we must reverse the motor and move until the home limit switch closes, stopping just past it.

In the latter case, we repeat the first move, because we always need to make the final approach to the home limit switch *from the same direction*, so that the final physical position is the same in either case!

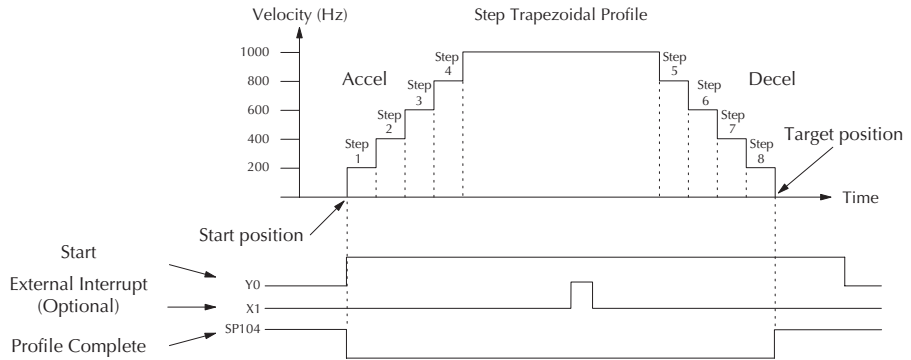




The home search profile will execute specific parts of the program, based on the order of detection of the limit switches. Ladder logic sets C0 to initiate a home search in the CW direction. If the CW limit is encountered, the program searches for home in the CCW direction, passes it slightly, and does the final CW search for home. After reaching home, the last ladder rung preloads the current position to "0".

## Step Trapezoidal Profile Operation

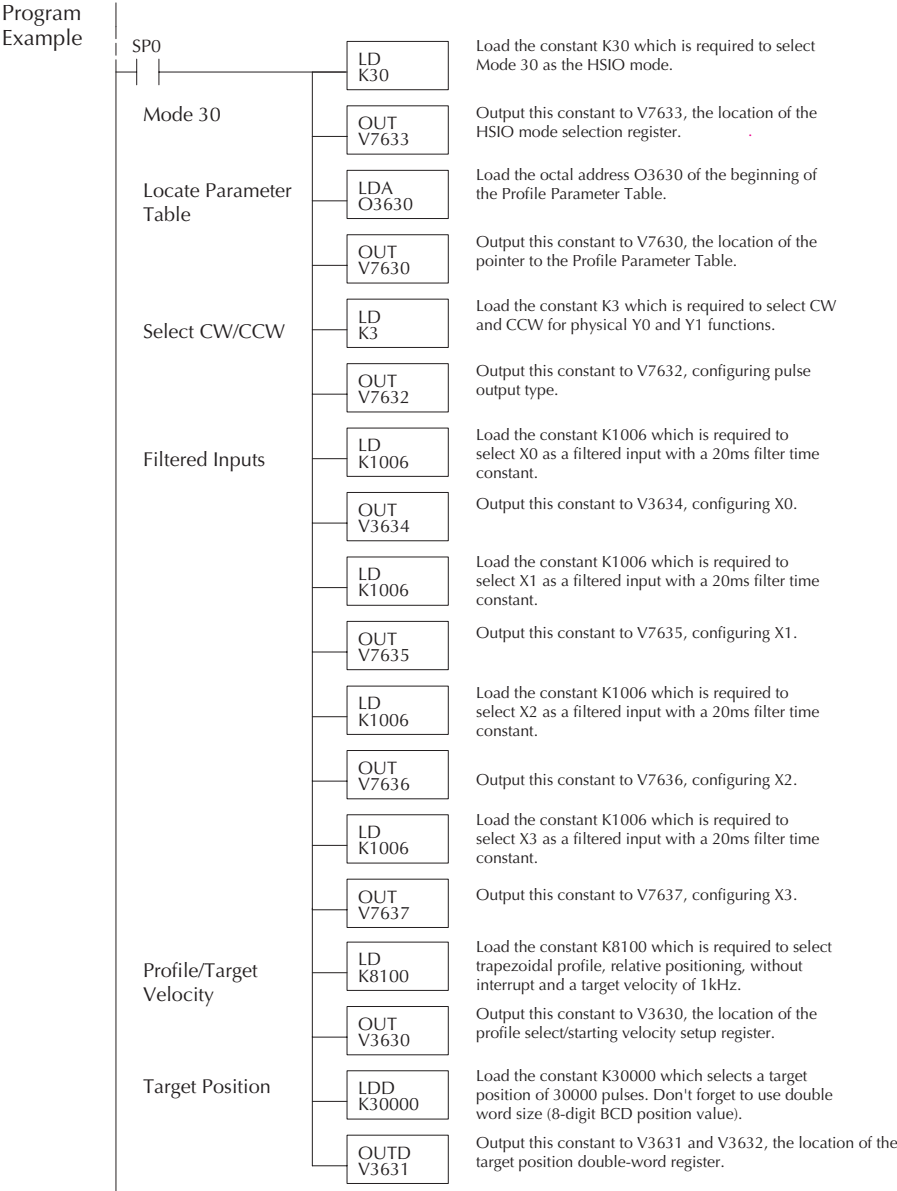
With this step trapezoidal profile, you can control the acceleration and deceleration slopes as you want.



The time line of signal traces below the profile indicates the order of events. The HSIO uses logical output Y0 as the start input to the HSIO, which starts the profile. Immediately, the HSIO turns off the Profile Complete signal (SP104), so the ladder program can monitor the progress of the move. Typically, a ladder program will monitor this bit so it knows when to initiate the next profile move. You can also use the external interrupt (X1). Once the external interrupt feature selected for the profile, the DL06 keeps outputting the pulses until X1 turns on. Then the DL06 outputs the pulses defined as the target position.

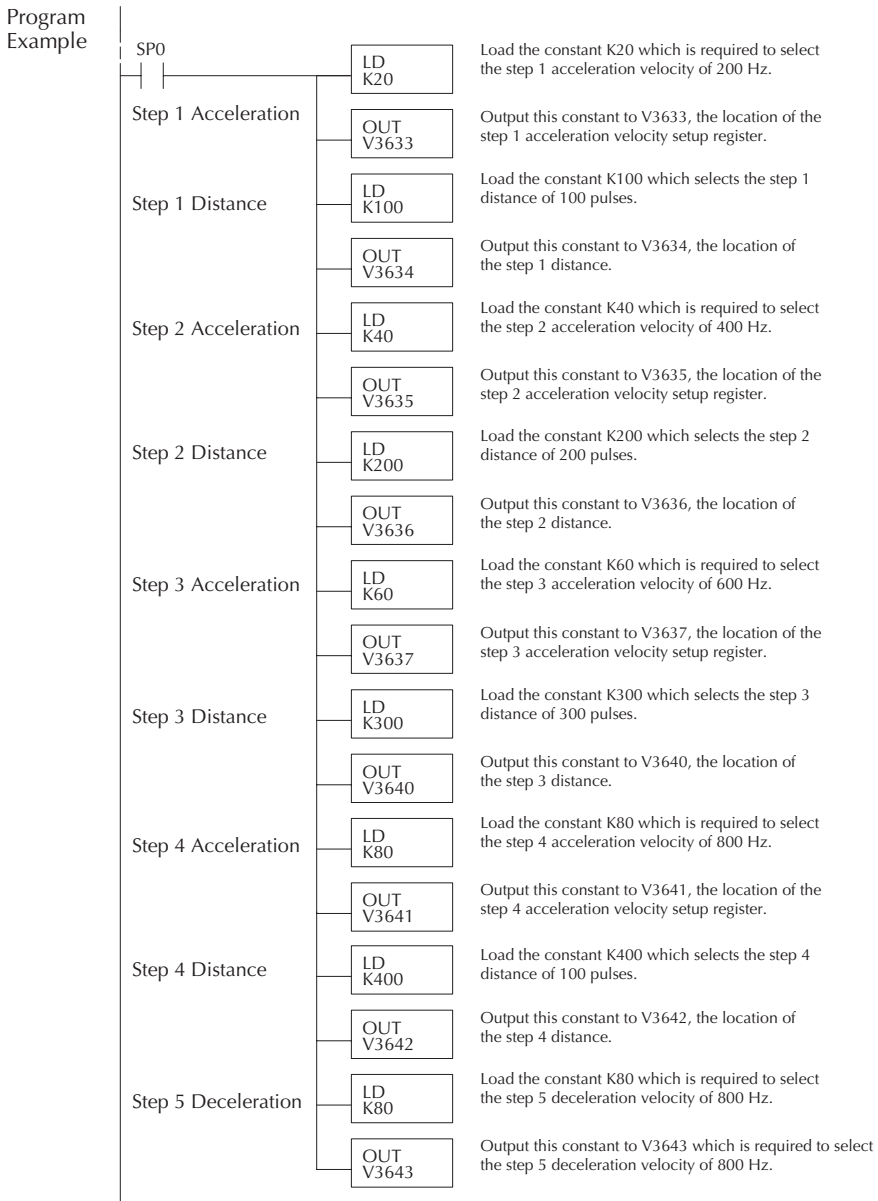
Each acceleration and deceleration slope consists of 4 steps. You can set up the velocity and the distance (number of pulses) of each step. You don't need to use all 4 steps of each slope. For instance, if you want to use only 2 steps, just set zero to the velocity and the distance of the 3rd and 4th step. If the acceleration slope and the deceleration slope are identical, you can just put zero into all the velocity and the distance parameters for the deceleration slope.

Program Example 4: Step Trapezoidal Profile



Continued on next page

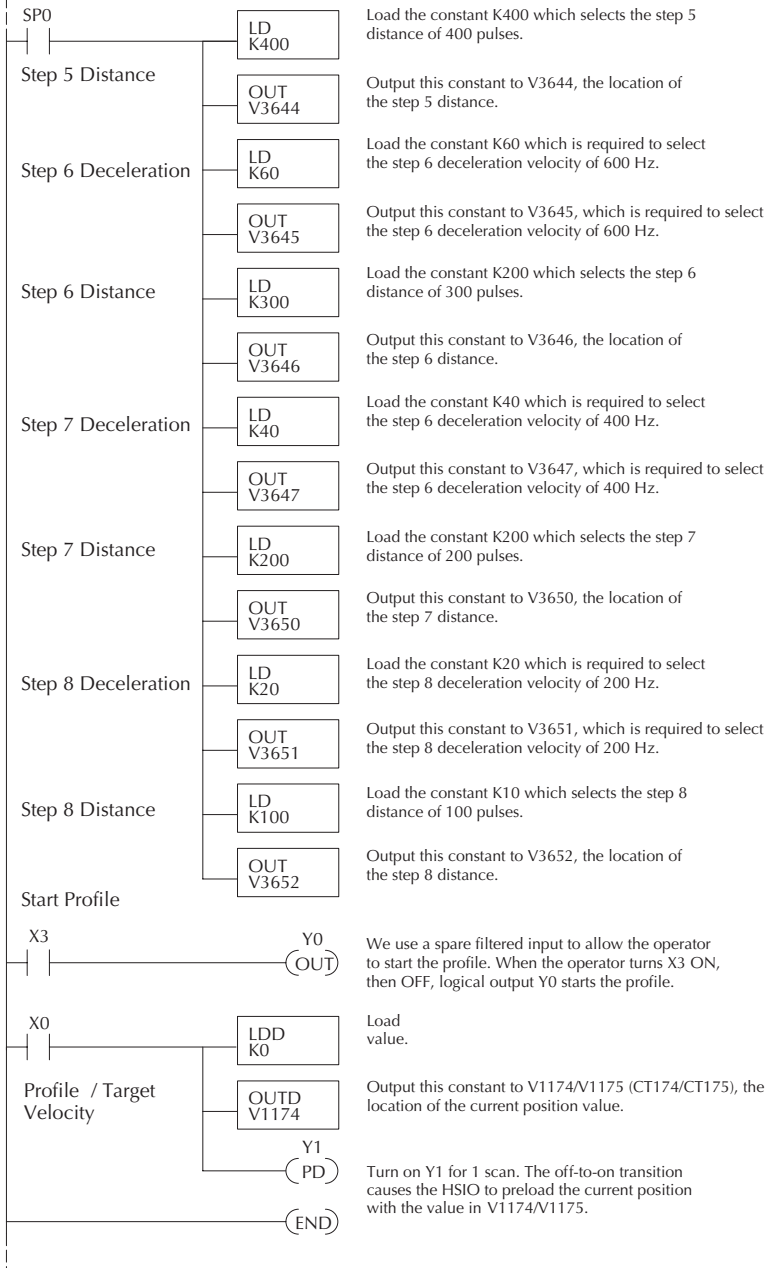
Continued from previous page



Continued on next page

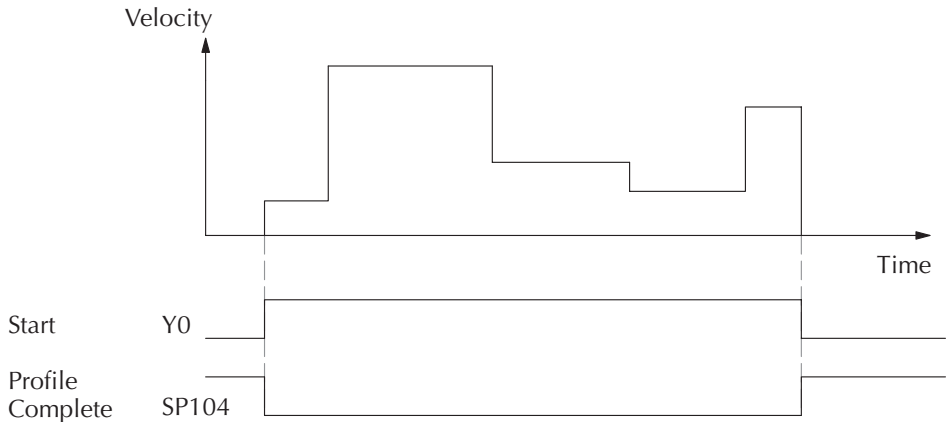
Continued from previous page

Program  
Example



## Velocity Profile Operation

The velocity profile is best suited for applications which involve motion but do not require moves to specific points. Conveyor speed control is a typical example.



The time line of signal traces below the profile indicates the order of events. Assuming the velocity is set greater than zero, motion begins when the Start input (Y0) energizes. Since there is no end position target, the profile is considered in progress as long as the Start input remains active. The profile complete logical input to ladder logic (X0) correlates directly to the Start input status when velocity profiles are in use.

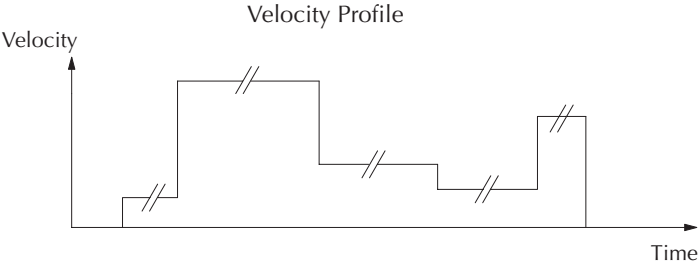
While the Start input is active, the ladder program can command a velocity change by writing a new value to the velocity register (V3633 by default). The full speed range of 40 Hz to 10 kHz is available. Notice from the drawing that there are no acceleration or deceleration ramps between velocity updates. This is how velocity profiling works with the HSIO. However, the ladder program can command more gradual velocity changes by incrementing or decrementing the velocity value more slowly. A counter or timer can be useful in creating your own acceleration/deceleration ramps. Unless the load must do a very complex move, it is easier to let the HSIO function generate the accel/decel ramps by selecting the trapezoidal or registration profiles instead.

Unlike the trapezoidal and registration profiles, you must specify the desired direction of travel with velocity profiles. Load the direction select register (V3631/V3632 by default) with 8000 0000 hex for CCW direction, or 0 for CW direction.

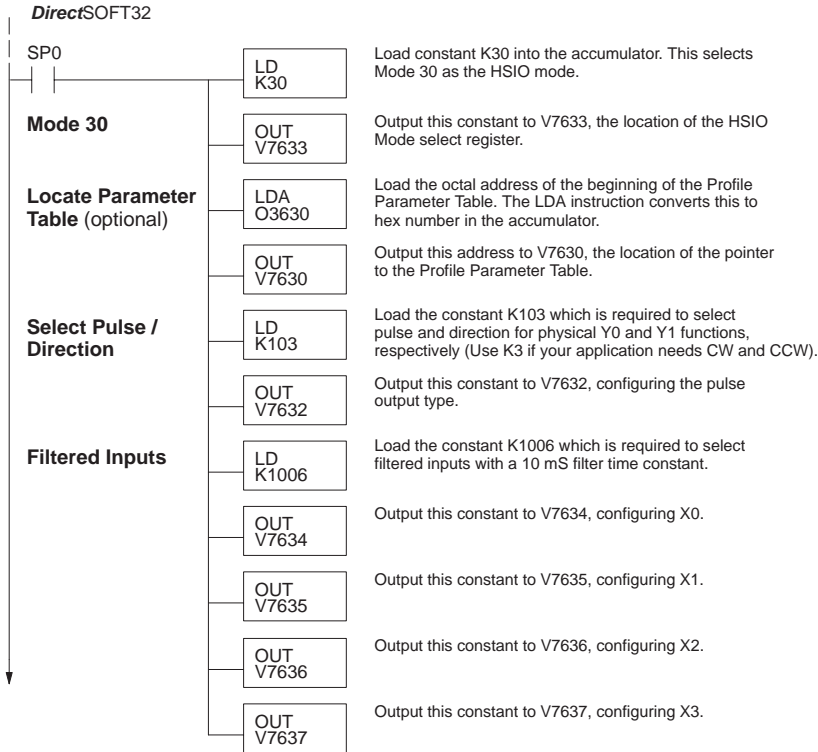


Program Example 5: Velocity Profile

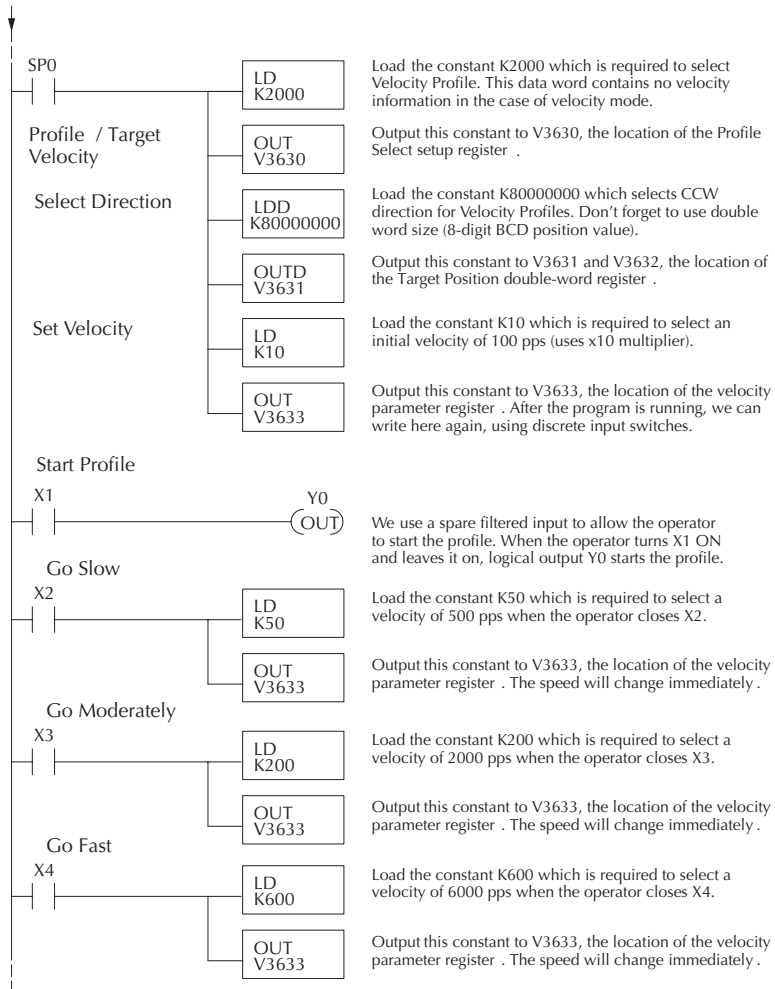
The velocity profile we want to perform is drawn and labeled in the following figure. Each velocity segment is of indefinite length. The velocity only changes when ladder logic (or other device writing to V-memory) updates the velocity parameter.



The following program uses dedicated discrete inputs to load in new velocity values. This program is fun to try, because you can create an infinite variety of profiles with just two or three input switches. The intent is to turn on only one of X2, X3, or X4 at a time. The beginning of the program contains all the necessary setup parameters for Pulse Output Mode 30. We only have to do this once in the program, so we use first-scan contact SP0 to trigger the setup.



## Program Example Cont'd



### Automatic Trapezoidal Profile Error Codes

The Profile Parameter Table starting at V3630 (default location) defines the profile. Certain numbers will result in an error when the HSIO attempts to use the parameters to execute a move profile. When an error occurs, the HSIO writes an error code in V3636.

Most errors can be corrected by rechecking the Profile Parameter Table values. The error is automatically cleared at powerup and at Program-to-Run Mode transitions.

Error Code	Error Description
0000	No error
0010	Requested profile type code is invalid (must use 4 to 6 or C to F)
0020	Target Velocity is not in BCD
0021	Target Velocity is specified to be less than 40 pps
0022	Target Velocity is specified to be greater than 10,000 pps
0030	Target Position value is not in BCD
0032	Direction Select is not 0 or 80000000.
0040	Starting Velocity is not in BCD
0041	Starting Velocity is specified to be less than 40 pps
0042	Starting Velocity is specified to be greater than 1,000 pps
0050	Acceleration Time is not in BCD
0051	Acceleration Time is zero
0052	Acceleration Time is greater than 10 seconds
0060	Deceleration Time is not in BCD
0061	Deceleration Time is zero
0062	Deceleration Time is greater than 10 seconds

### Troubleshooting Guide for Mode 30

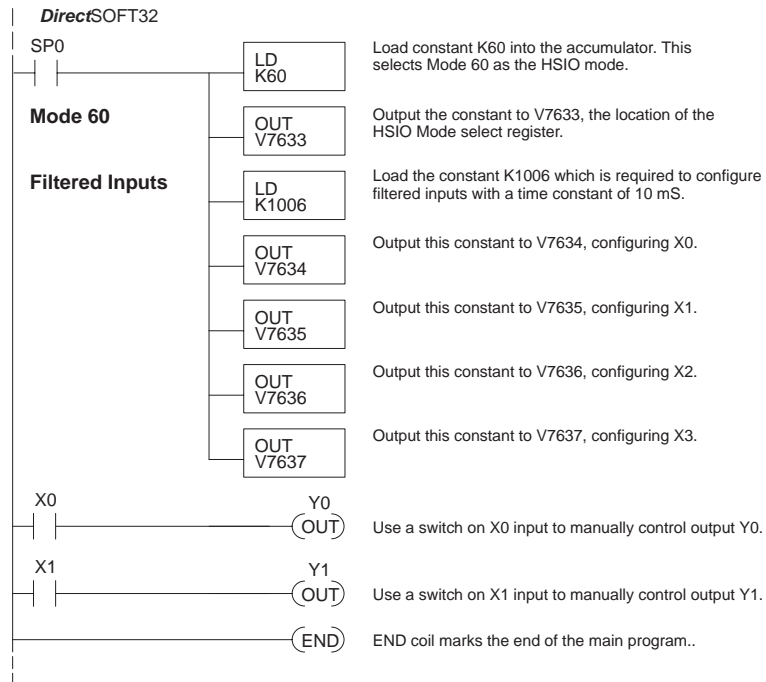
If you're having trouble with Mode 30 operation, please study the following symptoms and possible causes. The most common problems are listed below:

#### Symptom: The stepper motor does not rotate.

Possible causes:

1. **Configuration** – Verify that the HSIO actually generates pulses on outputs Y0 and Y1. Watch the status LEDs for Y0 and Y1 when you start a motion profile. If the LEDs flicker on and off or are steadily on, the configuration is probably correct.
2. **Programming error** – If there are no pulses on Y0 or Y1 you may have a programming error. Check the contents of V3636 for an error code that may be generated when the PLC attempts to do the move profile. Error code descriptions are given above.
3. **Check target value** – The profile will not pulse if the count value is equal to the target value (ex. count =0, target=0)

4. **Wiring** – Verify the wiring to the stepper motor is correct. Remember the signal ground connection from the PLC to the motion system is required.
5. **Motion system** – Verify that the drive is powered and enabled. To verify the motion system is working, you can use Mode 60 operation (normal PLC inputs/outputs) as shown in the test program below. With it, you can manually control Y0 and Y1 with X0 and X1, respectively. Using an input simulator is ideal for this type of manual debugging. With the switches you can single-step the motor in either direction. If the motor will not move with this simple control, Mode 30 operation will not be possible until the problem with the motor drive system or wiring is corrected.



6. **Memory Error** – HSI0 configuration parameters are stored in the CPU system memory. Corrupted data in this memory area can sometimes interfere with proper HSI0 operation. If all other corrective actions fail, initializing the scratchpad memory may solve the problem. With *DirectSOFT32*, select the *PLC* menu, then *Setup*, then *Initialize Scratchpad*.

## Symptom: The motor turns in the wrong direction.

Possible causes:

1. **Wiring** – If you have selected CW and CCW type operation, just swap the wires on Y0 and Y1 outputs.
2. **Direction control** – If you have selected Pulse and Direction type operation, just change the direction bit to the opposite state.

# Mode 40: High-Speed Interrupts

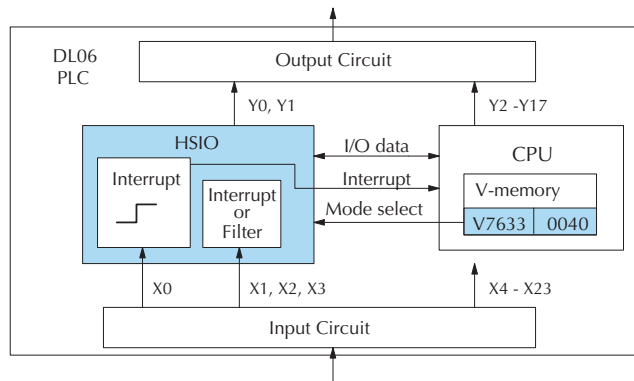
### Purpose

The HSIO Mode 40 provides a high-speed interrupt to the ladder program. This capability is provided for your choice of the following application scenarios:

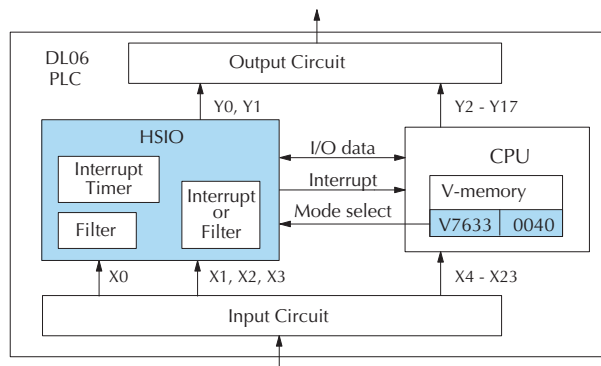
- External events need to trigger an interrupt subroutine in the CPU. Using immediate I/O instructions in the subroutine is typical.
- An interrupt routine needs to occur on a timed basis which is different from the CPU scan time (either faster or slower). The timed interrupt is programmable, from 5 to 999 mS.

### Functional Block Diagram

The HSIO circuit creates the high-speed interrupt to the CPU. The following diagram shows the external interrupt option, which uses X0. In this configuration X1, X2 and X3 are external interrupts or normal filtered inputs.

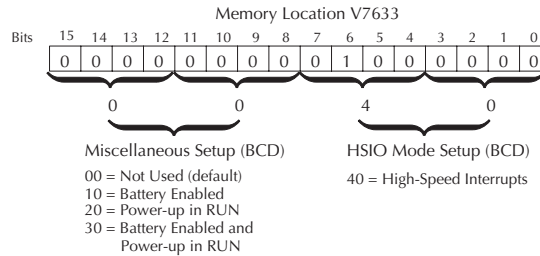


Alternately, you may configure the HSIO circuit to generate interrupts based on a timer, as shown below. In this configuration, inputs X0 is a filtered input.



## Setup for Mode 40

Recall that V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 40 in the lower byte of V7633 to select the High-Speed Counter Mode.



Choose the most convenient method of programming V7633 from the following:

- Include load and out instructions in your ladder program
- *DirectSOFT32's* memory editor
- Use the Handheld Programmer D2-HPP

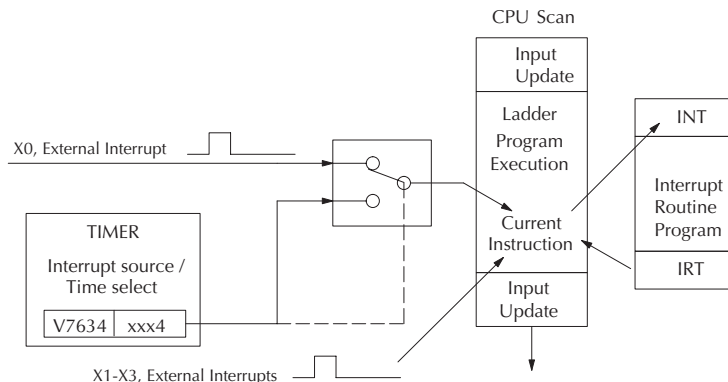
We recommend using the first method above so that the HSIO setup becomes an integral part of your application program. An example program later in this section shows how to do this.

## Interrupts and the Ladder Program

Refer to the drawing below. The source of the interrupt may be external (X0 - X3). An internal timer can be used instead of X0 as the interrupt source. The setup parameter in V7634 serves a dual purpose:

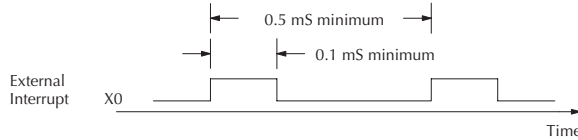
- It selects between the two interrupt sources (external or internal timer). The timed interrupt can only be used with X0.
- In the case of the timer interrupt, it programs the interrupt timebase between 5 and 999 mS.

The resulting interrupt uses label INT 0, 1, 2 or 3 in the ladder program. Be sure to include the Enable Interrupt (ENI) instruction at the beginning of your program. Otherwise, the interrupt routine will not be executed.



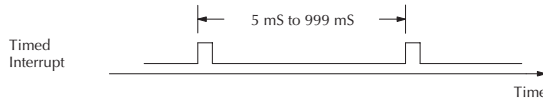
### External Interrupt Timing Parameters

External interrupt signals must meet certain timing criteria to guarantee an interrupt will result. Refer to the timing diagram below. The minimum pulse width is 0.1 mS. There must be some delay before the next interrupt pulse arrives, such that the interrupt period cannot be smaller than 0.5 mS.



### Timed Interrupt Parameters

When the timed interrupt is selected, the HSIO generates the interrupt to ladder logic. There is no interrupt “pulse width” in this case, but the interrupt period can be adjusted from 5 to 999 mS.



### X Input / Timed INT Configuration

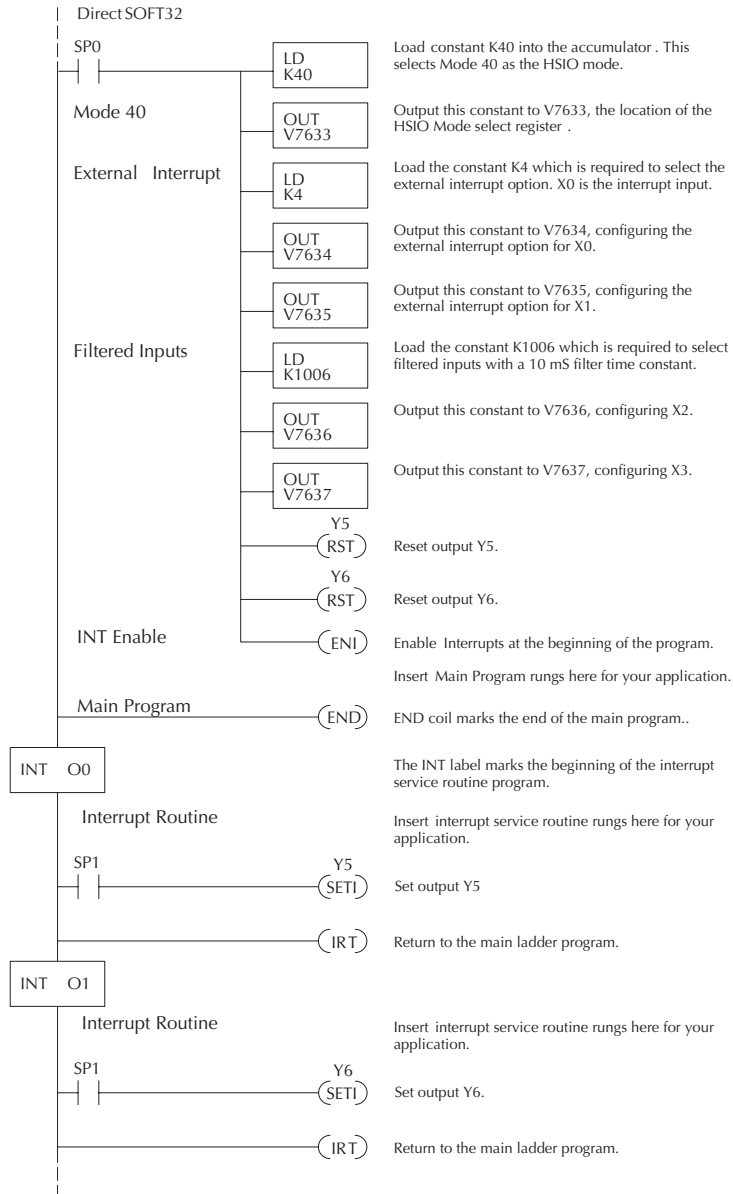
The configurable discrete input options for High-Speed Interrupt Mode are listed in the table below. Input X0 is the external interrupt when “0004” is in V7634. If you need a timed interrupt instead, then V7634 contains the interrupt time period, and input X0 becomes a filtered input (uses X1’s filter time constant by default). Inputs X0, X1, X2, and X3, can be filtered inputs, having individual configuration registers and filter time constants, interrupt inputs or counter inputs.

Input	Configuration Register	Function	Hex Code Required
X0	V7634	External Interrupt	0004 (default)
		Timed Interrupt	xxx4, xxx = INT timebase 5 - 999 ms (BCD)
X1	V7635	Interrupt	0004 (default)
		Pulse Input	0005
		Filtered Input	xx06 (xx = filter time) 0 - 99 ms (BCD)
X2	V7636	Interrupt	0004 (default)
		Pulse Input	0005
		Filtered Input	xx06 (xx = filter time) 0 - 99 ms (BCD)
X3	V7637	Interrupt	0004 (default)
		Pulse Input	0005
		Filtered Input	xx06 (xx = filter time) 0 - 99 ms (BCD)

If you are only using *one* of the points for an interrupt, you may want to select a different *main* mode (i.e. 10, 20, 30, 50, or 60); and then, just configure one of the terminals not taken as an interrupt. For example, you might want to configure your CPU for the UP counter mode (Mode 10) and use point 03 for a high speed interrupt. You should read the individual sections for any alternate mode you might choose. There you will find instructions on how to select a high speed interrupt as a secondary feature.

## Program Example 1: External Interrupt

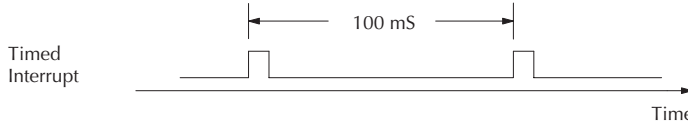
The following program selects Mode 40, then selects the external interrupt option for inputs X0 and X1. Inputs X2 and X3 are configured as filtered inputs with a 10 mS time constant. The program is otherwise generic, and may be adapted to your application.



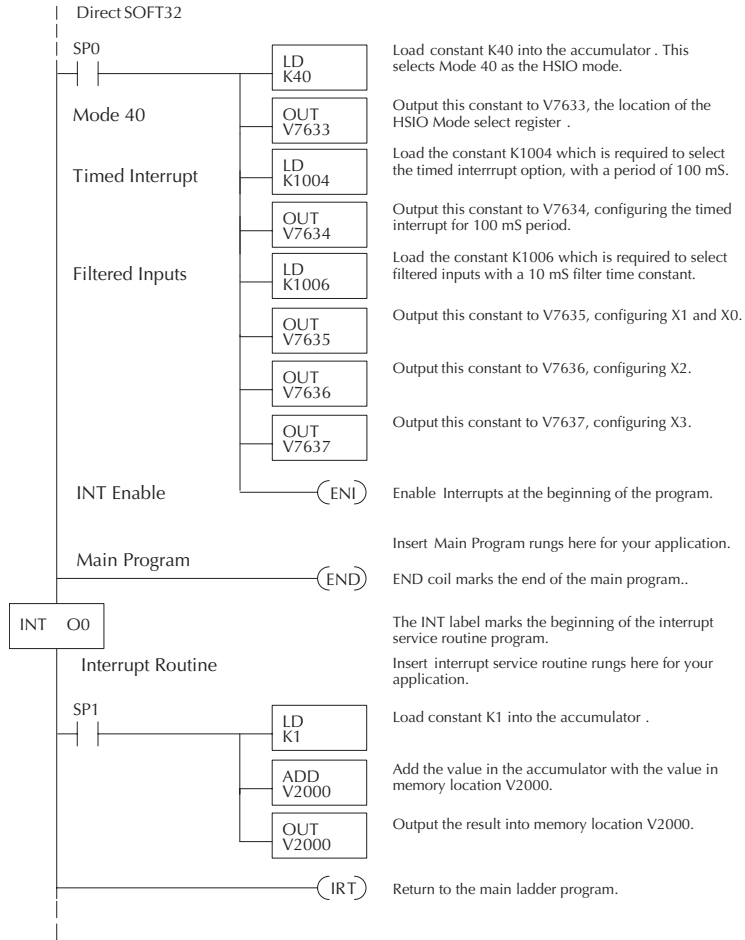


## Program Example 2: Timed Interrupt

The following program selects Mode 40, then selects the timed interrupt option, with an interrupt period of 100 mS.



Inputs X0, X1, X2, and X3, are configured as filtered inputs with a 10 mS time constant. Note that X0 uses the time constant from X1. The program is otherwise generic, and may be adapted to your application.



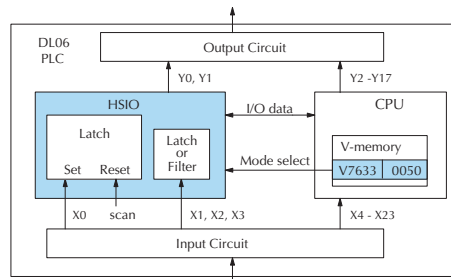
## Mode 50: Pulse Catch Input

### Purpose

The HSIO circuit has a pulse-catch mode of operation. It monitors the signal on inputs X0 - X3, preserving the occurrence of a narrow pulse. The purpose of the pulse catch mode is to enable the ladder program to “see” an input pulse which is shorter in duration than the current scan time. The HSIO circuit latches the input event on input X0 - X3 for one scan. This contact automatically goes off after one scan.

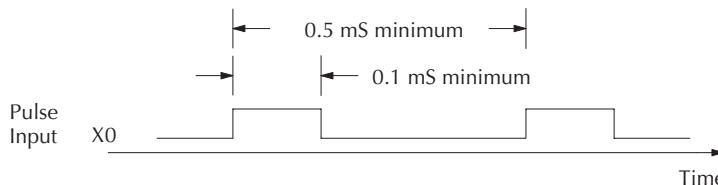
### Functional Block Diagram

Refer to the block diagram below. When the lower byte of HSIO Mode register V7633 contains a BCD “50”, the pulse catch mode in the HSIO circuit is enabled. X0 - X3 automatically become the pulse catch inputs, which set the latch on each rising edge. The HSIO resets the latch at the end of the next CPU scan. Inputs X1, X2, and X3 can be filtered discrete inputs, also.



### Pulse Catch Timing Parameters

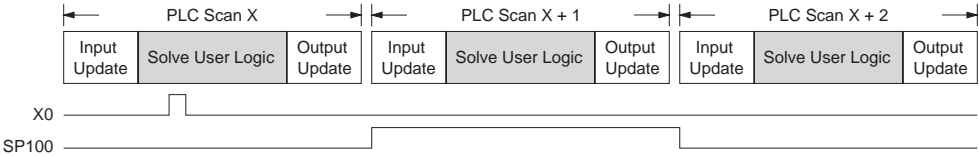
Signal pulses at X0 - X3 must meet certain timing criteria to guarantee a pulse capture will result. Refer to the timing diagram below. The input characteristics of X0 are fixed (it is not a programmable filtered input). The minimum pulse width is 0.1 mS. There must be some delay before the next pulse arrives, such that the pulse period cannot be smaller than 0.5 mS. If the pulse period is smaller than 0.5 mS, the next pulse will be considered part of the current pulse.



**Note:** that the pulse catch and filtered input functions are opposite in nature. The pulse catch feature seeks to capture narrow pulses, while the filter input feature seeks to reject narrow pulses.

## When to use Pulse Catch Mode

Use the pulse catch mode for applications where the input (e.g. X0) can not be used in the user program because the pulse width is very narrow. Use SP100 instead of X0. The SP100 contact stays on through the next scan, as shown above.

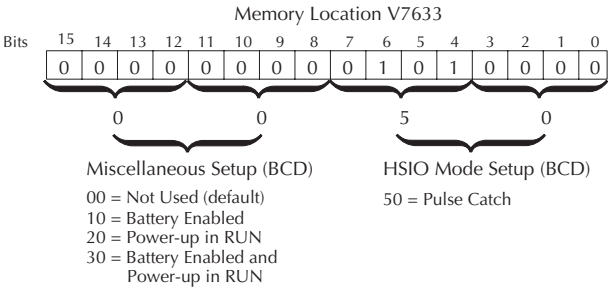


The status relay for X0 is SP100. The other status relays are shown in the table below.

Input	Status Relay
X0	SP100
X1	SP101
X2	SP102
X3	SP103

## Setup for Mode 50

Recall that V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 50 in the lower byte of V7633 to select the High-Speed Counter Mode.



Choose the most convenient method of programming V7633 from the following:

- Include load and out instructions in your ladder program
- *DirectSOFT32's* memory editor
- Use the Handheld Programmer D2-HPP

We recommend using the first method above so that the HSIO setup becomes an integral part of your application program. An example program later in this section shows how to do this.

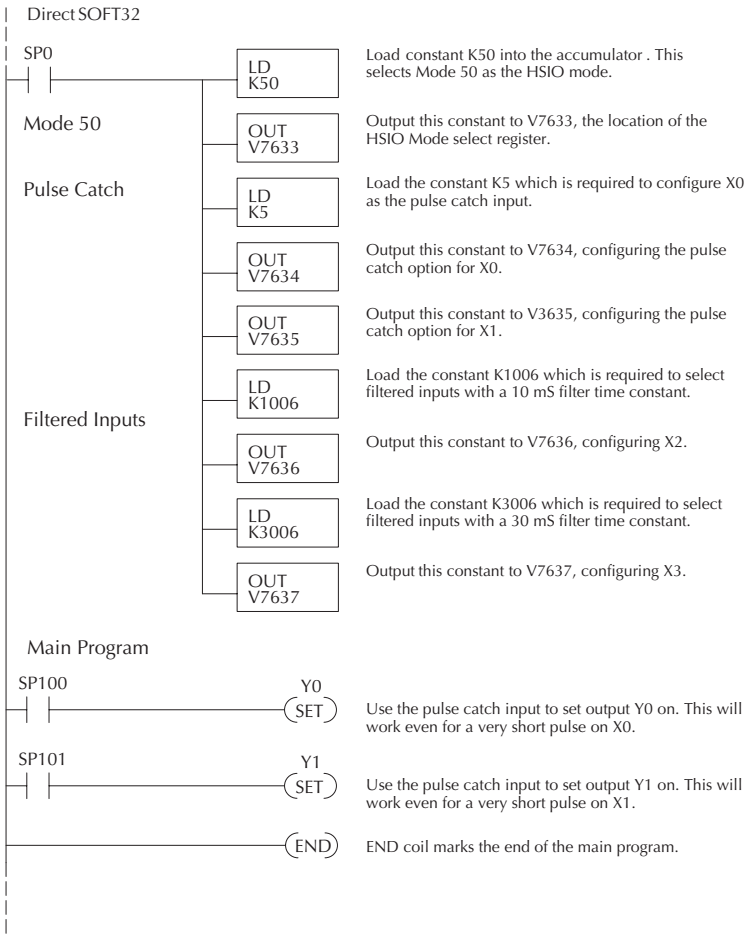
### X Input Configuration

The configurable discrete input options for Pulse Catch Mode are listed in the table below. Each input has its own configuration register and filter time constant.

Input	Configuration Register	Function	Hex Code Required
<b>X0</b>	V7634	Pulse Catch Input	0005 (default)
<b>X1</b>	V7635	Interrupt	0004
		Pulse Catch Input	0005 (default)
		Filtered Input	xx06 (xx = filter time) 0 - 99 ms (BCD)
<b>X2</b>	V7636	Interrupt	0004
		Pulse Catch Input	0005 (default)
		Filtered Input	xx06 (xx = filter time) 0 - 99 ms (BCD)
<b>X3</b>	V7637	Interrupt	0004
		Pulse Catch Input	0005 (default)
		Filtered Input	xx06 (xx = filter time) 0 - 99 ms (BCD)

## Program Example 1: Pulse Catch

The following program selects Mode 50, then programs the pulse catch code for X0 and X1. Inputs X2, and X3 are configured as filtered inputs with 10 and 30 mS time constants respectively. The program is otherwise generic, and may be adapted to your application.



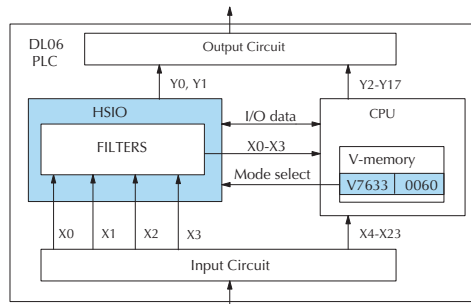
## Mode 60: Discrete Inputs with Filter

### Purpose

The last mode we will discuss for the HSIO circuit is Mode 60, Discrete Inputs with Filter. The purpose of this mode is to allow the input circuit to reject narrow pulses and accept wide ones, as viewed from the ladder program. This is useful in especially noisy environments or other applications where pulse width is important. In all other modes in this chapter, X0 to X3 usually support the mode functions as special inputs. Only spare inputs operate as filtered inputs by default. Now in Mode 60, all four inputs X0 through X3 function only as discrete filtered inputs.

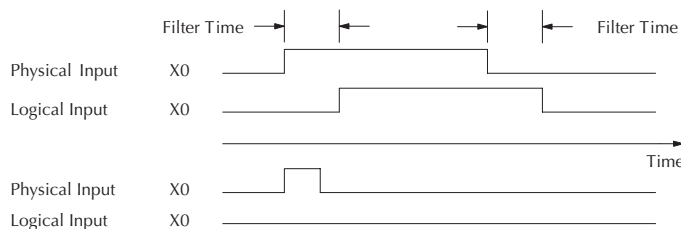
### Functional Block Diagram

Refer to the block diagram below. When the lower byte of HSIO Mode register V7633 contains a BCD “60”, the input filter in the HSIO circuit is enabled. Each input X0 through X3 has its own filter time constant. The filter circuit assigns the outputs of the filters as logical references X0 through X3.



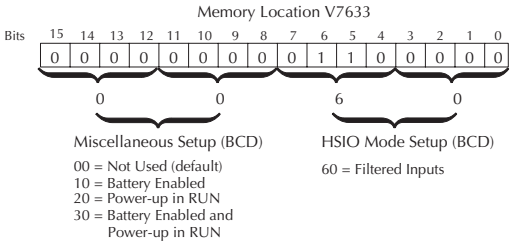
### Input Filter Timing Parameters

Signal pulses at inputs X0 – X3 are filtered by using a delay time. In the figure below, the input pulse on the top line is longer than the filter time. The resultant logical input to ladder is phase-shifted (delayed) by the filter time on both rising and falling edges. In the bottom waveforms, the physical input pulse width is smaller than the filter time. In this case, the logical input to the ladder program remains in the OFF state (input pulse was filtered out).



Setup for Mode 60

Recall that V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 60 in the lower byte of V7633 to select the High-Speed Counter Mode.



Choose the most convenient method of programming V7633 from the following:

- Include load and out instructions in your ladder program
- *DirectSOFT32*'s memory editor
- Use the Handheld Programmer D2-HPP

We recommend using the first method above so that the HSIO setup becomes an integral part of your application program. An example program later in this section shows how to to this.

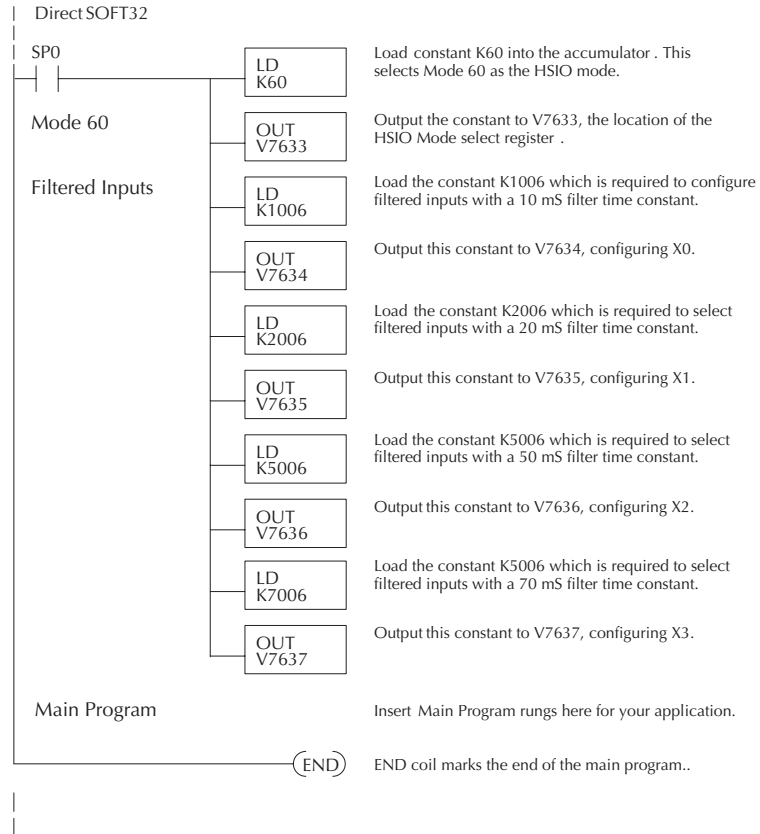
X Input Configuration

The configurable discrete input options for Discrete Filtered Inputs Mode are listed in the table below. The filter time constant (delay) is programmable from 0 to 99 mS (the input acts as a normal discrete input when the time constant is set to 0). The code for this selection occupies the upper byte of the configuration register in BCD. We combine this number with the required "06" in the lower byte to get "xx06", where xx = 0 to 99. Input X0, X1, X2, and X3 can only be filtered inputs. Each input has its own configuration register and filter time constant.

Input	Configuration Register	Function	Hex Code Required
X0	V7634	Filtered Input	xx06 (xx = filter delay time) 0 - 99 ms (BCD) (default)
X1	V7635	Filtered Input	xx06 (xx = filter delay time) 0 - 99 ms (BCD) (default)
X2	V7636	Filtered Input	xx06 (xx = filter delay time) 0 - 99 ms (BCD) (default)
X3	V7637	Filtered Input	xx06 (xx = filter delay time) 0 - 99 ms (BCD) (default)

## Program Example: Filtered Inputs

The following program selects Mode 60, then programs the filter delay time constants for inputs X0, X1, X2, and X3. Each filter time constant is different, for illustration purposes. The program is otherwise generic, and may be adapted to your application.





# CPU SPECIFICATIONS AND OPERATION

---



## CHAPTER 4

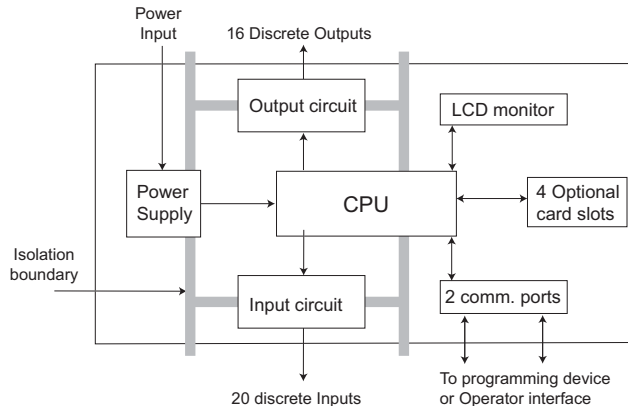
### In This Chapter

Introduction . . . . .	4-2
CPU Specifications . . . . .	4-3
CPU Hardware Setup . . . . .	4-4
Using Battery Backup . . . . .	4-8
CPU Operation . . . . .	4-12
I/O Response Time . . . . .	4-17
CPU Scan Time Considerations . . . . .	4-20
Memory Map . . . . .	4-25
DL06 System V-memory . . . . .	4-29
Control Relay Bit Map . . . . .	4-35
Timer Status Bit Map . . . . .	4-37
Counter Status Bit Map . . . . .	4-37
Remote I/O Bit Map . . . . .	4-38
Module Placement . . . . .	4-42
Power Budgeting . . . . .	4-44
Configuring the DL06's Comm Ports . . . . .	4-46
Connecting to MODBUS and DirectNET Networks . . . . .	4-48
MODBUS Port Configuration . . . . .	4-48
Non-Sequence Protocol (ASCII In/Out and PRINT) . . . . .	4-50
Network Slave Operation . . . . .	4-51
Network Master Operation . . . . .	4-56
Network Master Operation (using MRX and MWX Instructions) . . . . .	4-60

### Introduction

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

- Steps required to set up the CPU
- Operation of ladder programs
- Organization of Variable Memory



**Note:** The High-Speed I/O function (HSIO) consists of dedicated but configurable hardware in the DL06. It is not considered part of the CPU, because it does not execute the ladder program. For more on HSIO operation, see Chapter 3.

### DL06 CPU Features

The DL06 Micro PLC has 14.8K words of memory comprised of 7.6K of ladder memory and 7.6K words of V-memory (data registers). Program storage is in the FLASH memory which is a part of the CPU board in the PLC. In addition, there is RAM with the CPU which will store system parameters, V-memory, and other data which is not in the application program. The RAM is backed up by a “super-capacitor”, storing the data for several hours in the event of a power outage. The capacitor automatically charges during powered operation of the PLC.

The DL06 supports fixed I/O which includes twenty discrete input points and sixteen output points.

Over 220 different instructions are available for program development as well as extensive internal diagnostics that can be monitored from the application program or from an operator interface. Chapters 5, 6, and 7 provide detailed descriptions of the instructions.

The DL06 provides two built-in communication ports, so you can easily connect a handheld programmer, operator interface, or a personal computer without needing any additional hardware.

## CPU Specifications

Specifications	
Feature	DL06
Total Program memory (words)	14.8K
Ladder memory (words)	7680
Total V-memory (words)	7616
User V-memory (words)	7488
Non-volatile V Memory (words)	128
Contact execution (boolean)	2.0uS
Typical scan (1k boolean)	3 - 4mS
RLL Ladder style Programming	Yes
RLL and RLLPLUS Programming	Yes
Run Time Edits	Yes
Scan	Variable / fixed
Handheld programmer	Yes
<i>Direct</i> SOFT32 programming for Windows.	Yes
Built-in communication ports (RS232C)	Yes
FLASH Memory	Standard on CPU
Local Discrete I/O points available	36
Local Analog input / output channels maximum	None
High-Speed I/O (quad., pulse out, interrupt, pulse catch, etc.)	Yes, 2
I/O Point Density	20 inputs, 16 outputs
Number of instructions available (see Chapter 5 for details)	229
Control relays	1024
Special relays (system defined)	512
Stages in RLLPLUS	1024
Timers	256
Counters	128
Immediate I/O	Yes
Interrupt input (external / timed)	Yes
Subroutines	Yes
For/Next Loops	Yes
Math (Integer and floating point)	Yes
Drum Sequencer Instruction	Yes
Time of Day Clock/Calendar	Yes
Internal diagnostics	Yes
Password security	Yes
System error log	Yes
User error log	Yes
Battery backup	Optional D2-BAT-1 available (not included with unit)

## CPU Hardware Setup

### Communication Port Pinout Diagrams

Cables are available that allow you to quickly and easily connect a Handheld Programmer or a personal computer to the DL06 PLCs. However, if you need to build your own cables, use the pinout diagrams shown. The DL06 PLCs require an RJ-12 phone plug for port 1 and a 15-pin svga dsub for port 2.

The DL06 PLC has two built-in serial communication ports. Port 1 (RS232C only) is generally used for connecting to a D2-HPP, *Direct*SOFT32, operator interface, MODBUS slave only, or a DirectNET slave only. The baud rate is fixed at 9600 baud for port 1. Port 2 (RS232C/RS422/RS485) can be used to connect to a D2-HPP, *Direct*SOFT32, operator interface, MODBUS master/slave, DirectNET master/slave or ASCII in/out. Port 2 has a range of speeds from 300 baud to 38.4K baud.

Port 1 Pin Descriptions			Port 2 Pin Descriptions		
1	0V	Power (-) connection (GND)	1	5V	Power (+) connection
2	5V	Power (+) connection	2	TXD	Transmit data (RS-232C)
3	RXD	Receive data (RS-232C)	3	RXD	Receive data (RS-232C)
4	TXD	Transmit data (RS-232C)	4	RTS	Ready to send
5	5V	Power (+) connection	5	CTS	Clear to send
6	0V	Power (-) connection (GND)	6	RXD-	Receive data (-) (RS-422/485)

7	0V	Power (-) connection (GND)	7	TXD+	Transmit data (+) (RS-422/485)
8	0V	Power (-) connection (GND)	8	TXD-	Transmit data (-) (RS-422/485)
9	TXD+	Transmit data (+) (RS-422/485)	9	RTS+	Ready to send (+) (RS-422/485)
10	TXD-	Transmit data (-) (RS-422/485)	10	RTS-	Ready to send (-) (RS-422/485)
11	RTS+	Ready to send (+) (RS-422/485)	11	RXD+	Receive data (+) (RS-422/485)
12	RTS-	Ready to send (-) (RS-422/485)	12	CTS+	Clear to send (+) (RS-422/485)
13	RXD+	Receive data (+) (RS-422/485)	13	CTS-	Clear to send (-) (RS-422/485)
14	CTS+	Clear to send (+) (RS-422/485)	14		
15	CTS-	Clear to send (-) (RS-422/485)	15		

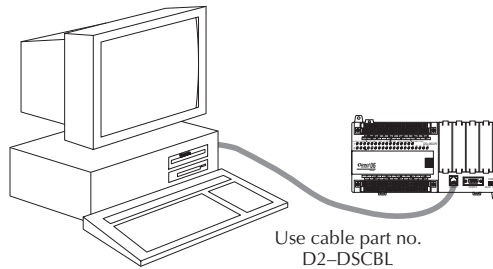
  

Communications Port 1	
Com 1	Connects to HPP, DirectSOFT32, operator interfaces, etc. 6-pin, RS232C Communication speed (baud): 9600 (fixed) Parity: odd (default) Station Address: 1 (fixed) 8 data bits 1 start, 1 stop bit Asynchronous, half-duplex, DTE Protocol: (auto-select) K-sequence (slave only), DirectNET (slave only), MODBUS (slave only)

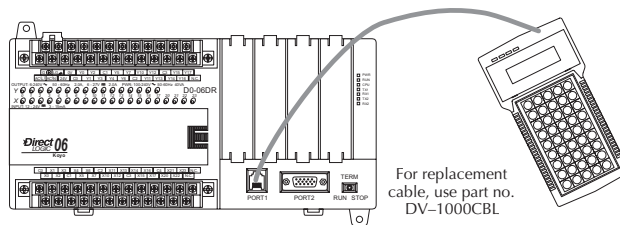
Communications Port 2	
Com 2	Connects to HPP, DirectSOFT32, operator interfaces, etc. 15-pin, multifunction port, RS232C, RS422, RS485 Communication speed (baud): 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 Parity: odd (default), even, none Station Address: 1 (default) 8 data bits 1 start, 1 stop bit Asynchronous, half-duplex, DTE Protocol: (auto-select) K-sequence (slave only), DirectNET (master/slave), MODBUS (master/slave), non-sequence/print/ASCII in/out

### Connecting the Programming Devices

If you're using a Personal Computer with the *DirectSOFT32™* programming package, you can connect the computer to either of the DL06's serial ports. For an engineering office environment (typical during program development), this is the preferred method of programming.



The Handheld programmer D2-HPP is connected to the CPU with a handheld programmer cable. This device is ideal for maintaining existing installations or making small program changes. The handheld programmer is shipped with a cable, which is approximately 6.5 feet (200 cm) long.

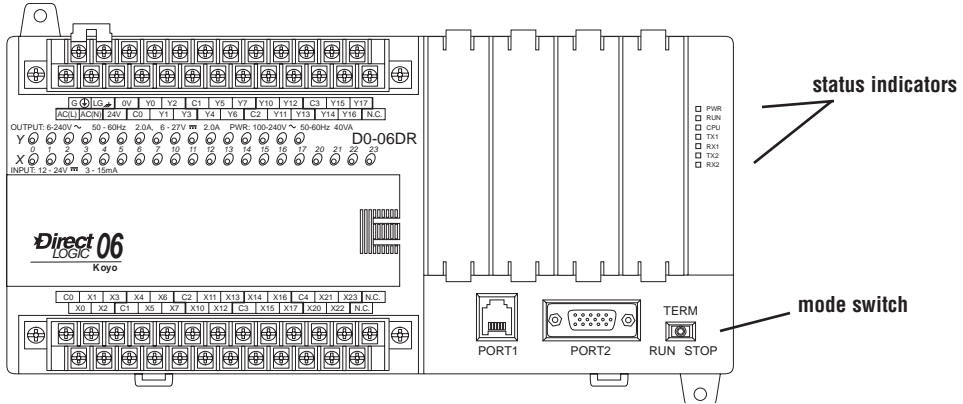


### CPU Setup Information

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

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

The following paragraphs provide the setup information necessary to get the CPU ready for programming. They include setup instructions for either type of programming device you are using. The D2-HPP Handheld Programmer Manual provides the Handheld keystrokes required to perform all of these operations. The *DirectSOFT32™* Manual provides a description of the menus and keystrokes required to perform the setup procedures via *DirectSOFT32*.



## Status Indicators

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

## Mode Switch Functions

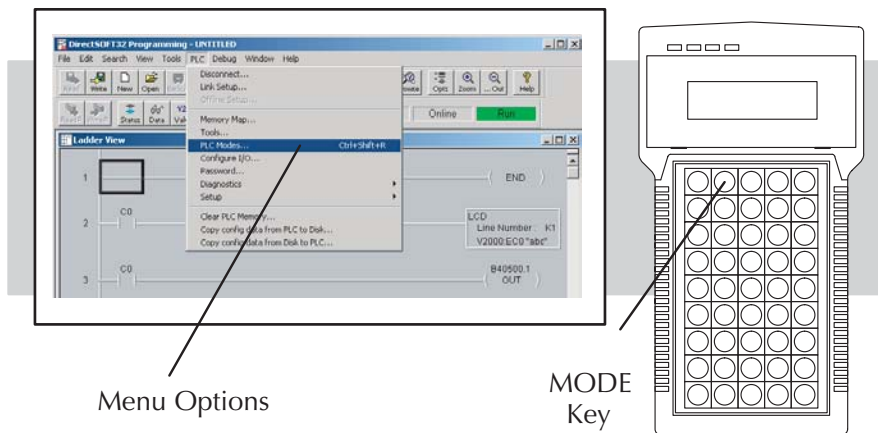
The mode switch on the DL06 PLC provides positions for enabling and disabling program changes in the CPU. Unless the mode switch is in the TERM position, RUN and STOP mode changes will not be allowed by any interface device, (handheld programmer, *DirectSOFT32* programming package or operator interface). 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.

Indicator	Status	Meaning
PWR	ON	Power good
	OFF	Power failure
RUN	ON	CPU is in Run Mode
	OFF	CPU is in Stop or program Mode
	Blinking	CPU is in firmware update mode
CPU	ON	CPU self diagnostics error
	OFF	CPU self diagnostics good
	Blinking	Low battery
TX1	ON	Data is being transmitted by the CPU - Port 1
	OFF	No data is being transmitted by the CPU - Port 1
RX1	ON	Data is being received by the CPU - Port 1
	OFF	No data is being received by the CPU - Port 1
TX2	ON	Data is being transmitted by the CPU - Port 2
	OFF	No data is being transmitted by the CPU - Port 2
RX2	ON	Data is being received by the CPU - Port 2
	OFF	No data is being received by the CPU - Port 2

## Changing Modes in the DL06 PLC

Mode Switch Position	CPU Action
<b>RUN (Run Program)</b>	CPU is forced into the RUN mode if no errors are encountered. No changes are allowed by the attached programming/monitoring device.
<b>TERM (Terminal) RUN</b>	PROGRAM and the TEST modes are available. Mode and program changes are allowed by the programming/monitoring device.
<b>STOP</b>	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. You can use the CPU mode switch to select the operating mode, or you can place the mode switch in the TERM position and use a programming device to change operating modes. With the switch in this position, the CPU can be changed between Run and Program modes. You can use either *DirectSOFT32* or the Handheld Programmer to change the CPU mode of operation. With *DirectSOFT32* you use a menu option in the PLC menu. With the Handheld Programmer, you use the MODE key.



### Mode of Operation at Power-up

The DL06 CPU will normally power-up in the mode that it was in just prior to the power interruption. For example, if the CPU was in Program Mode when the power was disconnected, the CPU will power-up in Program Mode (see warning note below).



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

### Using Battery Backup

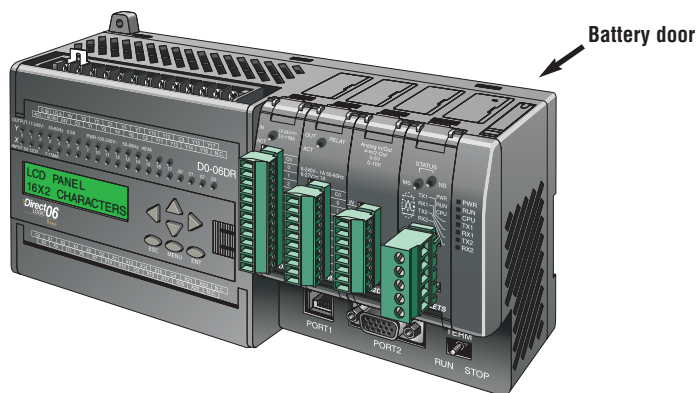
An optional lithium battery is available to maintain the system RAM retentive memory when the DL06 system is without external power. Typical CPU battery life is five years, which includes PLC runtime and normal shutdown periods. However, consider installing a fresh battery if your battery has not been changed recently and the system will be shut down for a period of more than ten days.



**NOTE:** Before installing or replacing your CPU battery, back-up your V-memory and system parameters. You can do this by using **DirectSOFT32** to save the program, V-memory, and system parameters to hard/floppy disk on a personal computer.

#### To install the D2-BAT-1 CPU battery in the DL06 CPU:

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



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

#### Enabling the Battery Backup

Setting bit 12 in V7633 ON enables the battery circuit (SET B7633.12). In this mode the CPU will maintain the data in C, S, T, CT, and V memory when power is removed from the CPU, provided the battery is good. The use of a battery can also determine which operating mode is entered when the system power is connected.

Whenever you have installed a battery, the battery always backs up the memory. However, the low battery indication can be disabled by turning off bit 12 in V7633.



## Auxiliary Functions

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

You can access the AUX Functions from *DirectSOFT32* or from the D2-HPP Handheld Programmer. The manuals for those products provide step-by-step procedures for accessing the AUX Functions. Some of these AUX Functions are designed specifically for the Handheld Programmer setup, so they will not be needed (or available) with the *DirectSOFT32* package. The following table shows a list of the Auxiliary functions for the Handheld Programmer.

AUX 2* — RLL Operations	
21	Check Program
22	Change Reference
23	Clear Ladder Range
24	Clear All Ladders
AUX 3* — V-Memory Operations	
31	Clear V Memory
AUX 4* — I/O Configuration	
41	Show I/O Configuration
42	I/O Diagnostics
44	Power Up I/O Configuration check
45	Select Configuration
46	Configure I/O
AUX 5* — CPU Configuration	
51	Modify Program Name
52	Display/Change Calendar
53	Display Scan Time
54	Initialize Scratchpad
55	Set Watchdog Timer
56	Set Communication Port 2
57	Set Retentive Ranges

58	Test Operations
59	Override Setup
5B	HSIO Configuration
5C	Display Error History
5D	Scan Control Setup
AUX 6* — Handheld Programmer Configuration	
61	Show Revision Numbers
62	Beeper On / Off
65	Run Self Diagnostics
AUX 7* — EEPROM Operations	
71	Copy CPU memory to HPP EEPROM
72	Write HPP EEPROM to CPU
73	Compare CPU to HPP EEPROM
74	Blank Check (HPP EEPROM)
75	Erase HPP EEPROM
76	Show EEPROM Type (CPU and HPP)
AUX 8* — Password Operations	
81	Modify Password
82	Unlock CPU
83	Lock CPU

## Clearing an Existing Program

Before you enter a new program, be sure to always clear ladder memory. You can use AUX Function 24 to clear the complete program. You can also use other AUX functions to clear other memory areas.

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

## Initializing System Memory

The DL06 Micro PLC maintains 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 in system memory. 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 load in new programs without ever initializing system memory.

Remember, this AUX function will reset all system memory. If you have set special parameters such as retentive ranges, etc. they will be erased when AUX 54 is used. Make sure that you have considered all ramifications of this operation before you select it.

## Setting Retentive Memory Ranges

The DL06 PLCs 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	DL06	
	Default Range	Available Range
Control Relays	C1000 – C1777	C0 – C1777
V Memory	V400 – V37777	V0 – V37777
Timers	None by default	T0 – T377
Counters	CT0 – CT177	CT0 – CT177
Stages	None by default	S0 – S1777

You can use AUX 57 to set the retentive ranges. You can also use DirectSOFT32. menus to select the retentive ranges. Appendix A contains detailed information about auxiliary functions.



**WARNING:** The DL06 CPUs do not come with a battery. The super capacitor will retain the values in the event of a power loss, but only for a short period of time, depending on conditions. If the retentive ranges are important for your application, make sure you obtain the optional battery.

## Using a Password

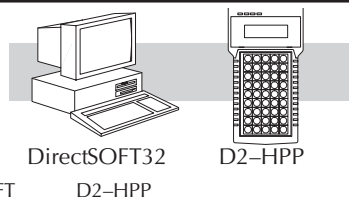
The DL06 PLCs allow you to use a password to help minimize the risk of unauthorized program and/or data changes. Once you enter a password you can “lock” the PLC against access. Once the CPU is locked you must enter the password before you can use a programming device to change any system parameters.

You can select an 8-digit numeric password. The Micro PLCs are shipped from the factory with a password of 00000000. All zeros removes the password protection. If a password has been entered into the CPU you cannot just enter all zeros to remove it. Once you enter the correct password, you can change the password to all zeros to remove the password protection.



**WARNING:** Make sure you remember your password. If you forget your password you will not be able to access the CPU. The Micro PLC must be returned to the factory to have the password (along with the ladder project) removed.

You can use the D2-HPP Handheld Programmer or *DirectSOFT32™*. to enter a password. The following diagram shows how you can enter a password with the Handheld Programmer.



Select AUX 81



PASSWORD  
00000000

Enter the new 8-digit password



PASSWORD  
XXXXXXXX

There are three ways to lock the CPU once the password has been entered.

1. If the CPU power is disconnected, the CPU will be automatically locked against access.
2. If you enter the password with *DirectSOFT32*, the CPU will be automatically locked against access when you exit *DirectSOFT32*.
3. Use AUX 83 to lock the CPU.

When you use *DirectSOFT32*, you will be prompted for a password if the CPU has been locked. If you use the Handheld Programmer, you have to use AUX 82 to unlock the CPU. Once you enter AUX 82, you will be prompted to enter the password.



*Note: The DL06 CPUs support multi-level password protection of the ladder program. This allows password protection while not locking the communication port to an operator interface. The multi-level password can be invoked by creating a password with an upper case “A” followed by seven numeric characters (e.g. A1234567).*

### CPU Operation

Achieving the proper control for your equipment or process requires a good understanding of how DL06 CPUs control all aspects of system operation. There are four main areas to understand before you create your application program:

- CPU Operating System — the CPU manages all aspects of system control. A quick overview of all the steps is provided in the next section.
- CPU Operating Modes — The two primary modes of operation are Program Mode and Run Mode.
- CPU Timing — The two important areas we discuss are the I/O response time and the CPU scan time.
- CPU Memory Map — DL06 CPUs offer a wide variety of resources, such as timers, counters, inputs, etc. The memory map section shows the organization and availability of these data types.

#### CPU Operating System

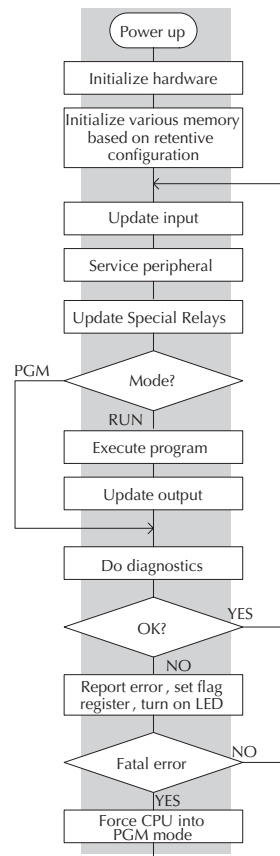
At powerup, the CPU initializes the internal electronic hardware. Memory initialization starts with examining the retentive memory settings. In general, the contents of retentive memory is

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.

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

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

You can use a programming device, such as *DirectSOFT32* or the D2-HPP Handheld Programmer to place the CPU in Program Mode.

## Run Mode

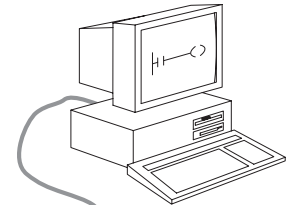
In Run Mode, the CPU executes the application program and updates the I/O system. You can perform many operations during Run Mode. Some of these include:

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

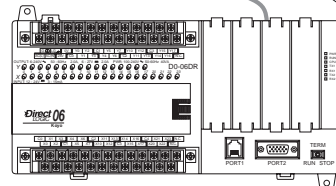
Run Mode operation can be divided into several key areas. For the vast majority of applications, some of these execution segments are more important than others. For example, you need to understand how the CPU updates the I/O points, handles forcing operations, and solves the application program. The remaining segments are not that important for most applications.

You can use *DirectSOFT32* or the D2-HPP Handheld Programmer to place the CPU in Run Mode.

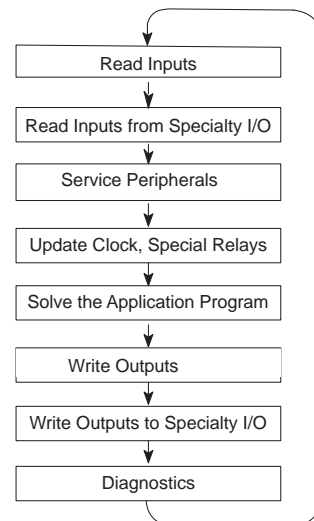
You can also edit the program during Run Mode. The Run Mode Edits are not “bumpless” to the outputs. Instead, the CPU maintains the outputs in their last state while it accepts the new program information. If an error is found in the new program, then the CPU will turn all the outputs off and enter the Program Mode. This feature is discussed in more detail in Chapter 9.



Download Program



Normal Run mode scan



**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 the 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.

### Service Peripherals and Force I/O

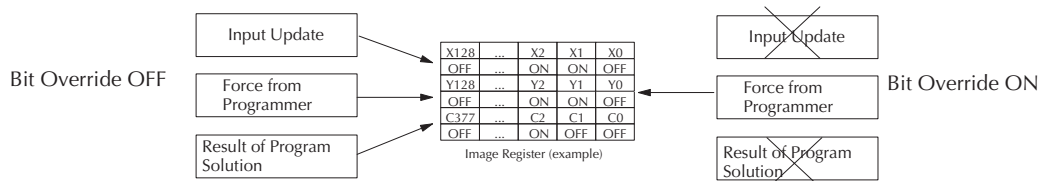
After the CPU reads the inputs from the input modules, it reads any attached peripheral devices. This is primarily a communications service for any attached devices. For example, it would read a programming device to see if any input, output, or other memory type status needs to be modified. There are two basic types of forcing available with the DL06 CPUs:

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

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

**Bit Override** — Bit override can be enabled on a point-by-point basis by using AUX 59 from the Handheld Programmer or, by a menu option from within *DirectSOFT32™*. Bit override basically disables any changes to the discrete point by the CPU. For example, if you enable bit override for X1, and X1 is off at the time, then the CPU will not change the state of X1. This means that even if X1 comes on, the CPU will not acknowledge the change. So, if you used X1 in the program, it would always be evaluated as “off” in this case. Of course, if X1 was on when the bit override was enabled, then X1 would always be evaluated as “on”.

There is an advantage available when you use the bit override feature. The regular forcing is not disabled because the bit override is enabled. For example, if you enabled the Bit Override for Y0 and it was off at the time, then the CPU would not change the state of Y0. However, you can still use a programming device to change the status. Now, if you use the programming device to force Y0 on, it will remain on and the CPU will not change the state of Y0. If you then force Y0 off, the CPU will maintain Y0 as off. The CPU will never update the point with the results from the application program or from the I/O update until the bit override is removed. The following diagram shows a brief overview of the bit override feature. Notice the CPU does not update the Image Register when bit override is enabled.



**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.

## CPU Bus Communication

It is possible to transfer data to and from the CPU over the CPU bus on the backplane. This data is more than standard I/O point status. This type of communications can only occur on the CPU (local) base. 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 DL06 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

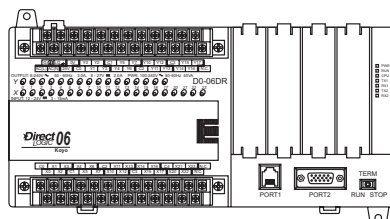
The CPU evaluates each instruction in the application program during this segment of the scan cycle. The instructions define the relationship between the input conditions and the desired output response. The CPU uses the output image register area to store the status of the desired action for the outputs.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

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

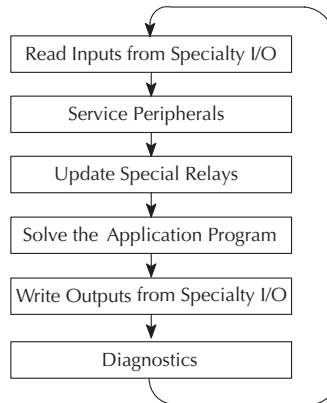
You may recall that you can force various types of points in the system. (This was discussed earlier in this chapter.) If any I/O points or memory data have been forced, the output image register also contains this information.

### Solve PID Loop Equations

The DL06 CPU can process up to 8 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.



Normal Run mode scan





## **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. Remember, the CPU also made sure that 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 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.

## **Diagnostics**

During this part of the scan, the CPU performs all system diagnostics and other tasks such as calculating the scan time and resetting the watchdog timer. There are many different error conditions that are automatically detected and reported by the DL06 PLCs. Appendix B contains a listing of the various error codes.

Probably one of the more important things that occurs during this segment is the scan time calculation and watchdog timer control. The DL06 CPU has a “watchdog” timer that stores the maximum time allowed for the CPU to complete the solve application segment of the scan cycle. If this time is exceeded the CPU will enter the Program Mode and turn off all outputs. The default value set from the factory is 200 ms. An error is automatically reported. For example, the Handheld Programmer would display the following message “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.

# **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 in such a short period of time that you may never have to concern yourself with the aspects of system timing. However, some applications do require extremely fast update times. In these cases, you may need to know how to determine the amount of time spent during the various segments of operation.

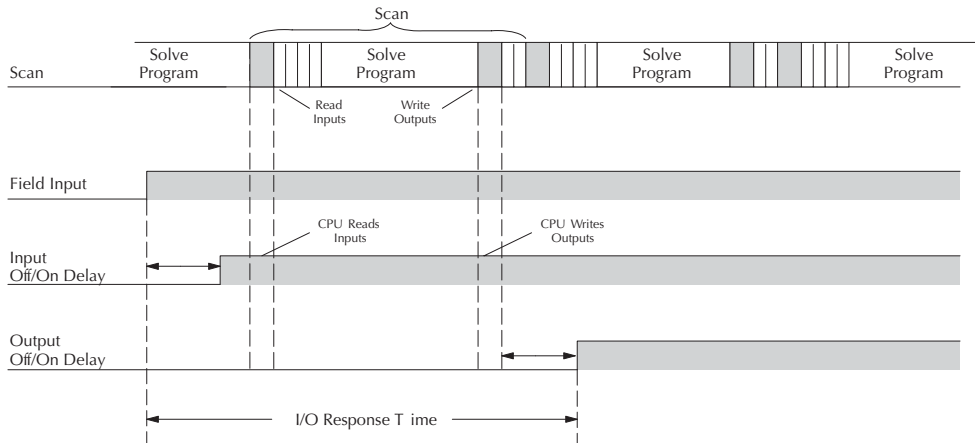
There are four things that can affect the I/O response time.

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

The next paragraphs show how these items interact to affect the response time.

### Normal Minimum I/O Response

The I/O response time is shortest when the input changes 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.

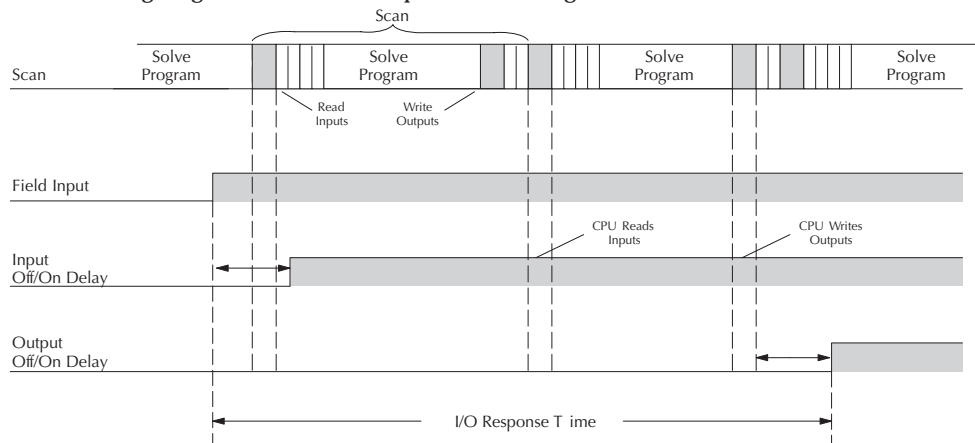


In this case, you can calculate the response time by simply adding the following items:

$$\text{Input Delay} + \text{Scan Time} + \text{Output Delay} = \text{Response Time}$$

### Normal Maximum I/O Response

The I/O response time is longest when the input changes just after the Read Inputs portion of the execution cycle. In this case the new input status is not 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:

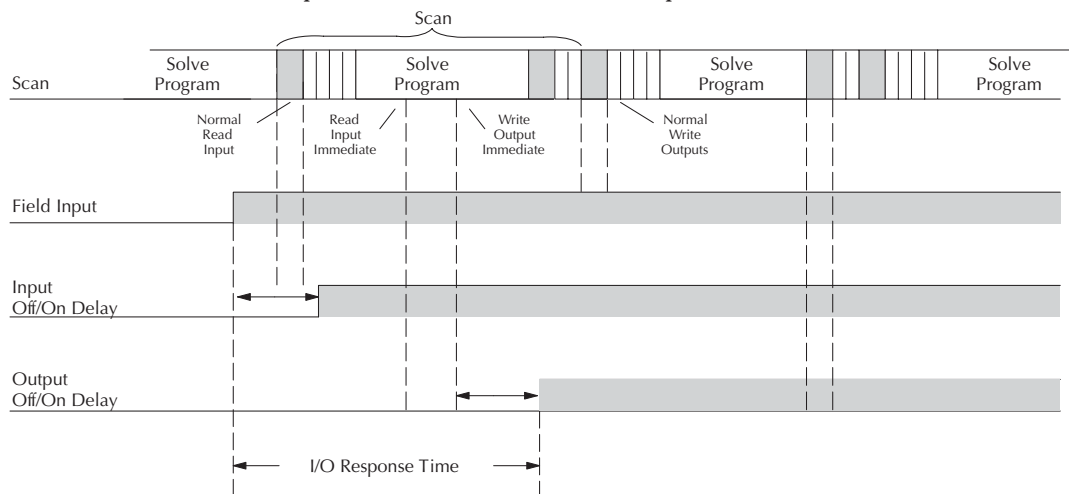
$$\text{Input Delay} + (2 \times \text{Scan Time}) + \text{Output Delay} = \text{Response Time}$$

## Improving Response Time

There are a few things you can do to help improve throughput.

- You can choose instructions with faster execution times
- You can use immediate I/O instructions (which update the I/O points during the program execution)
- You can use the HSIO Mode 50 Pulse Catch features designed to operate in high-speed environments. See Chapter 3 for details on using this feature.

Of these three things the Immediate I/O instructions are probably the most important and most useful. The following example shows how an immediate input instruction and immediate output instruction would affect the response time.



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

$$\text{Input Delay} + \text{Instruction Execution Time} + \text{Output Delay} = \text{Response Time}$$

The instruction execution time would be calculated by adding the time for the immediate input instruction, the immediate output instruction, and any other instructions in between the two.



**NOTE:** Even though the immediate instruction reads the most current status from I/O, it only uses the results to solve that one instruction. It does not use the new status to update the image register. Therefore, any regular instructions that follow will still use the image register values. Any immediate instructions that follow will access the I/O again to update the status.

### CPU Scan Time Considerations

The scan time covers all the cyclical tasks that are performed by the operating system. You can use *DirectSOFT32*, 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 following are the most important:

- Input Update
- Peripheral Service
- Program Execution
- Output Update
- Timed Interrupt Execution

The one you 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 type and peripheral devices can also affect the scan time. However, these things are usually dictated by the application.

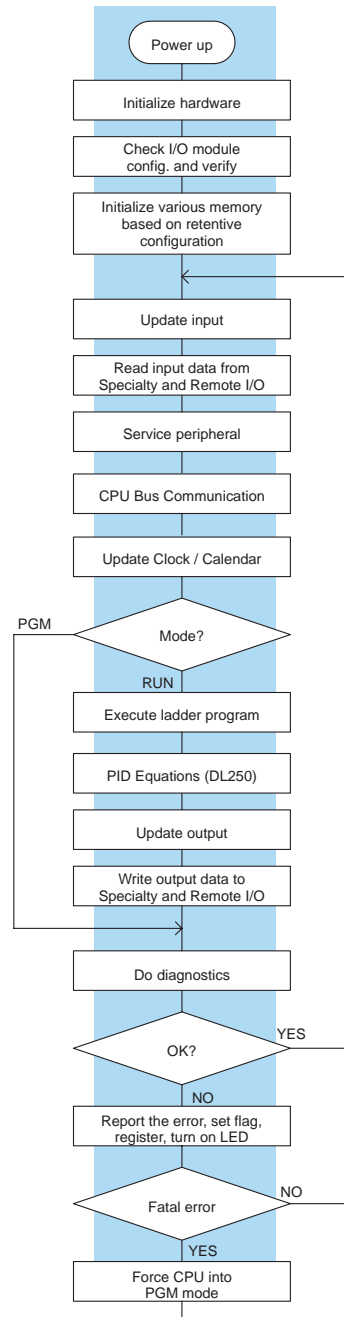
The following paragraphs provide some general information on how much time some of the segments can require.

#### Reading Inputs

The time required during each scan to read the input status of built-in inputs is 52.6  $\mu$ s. Don't confuse this with the I/O response time that was discussed earlier.

#### Writing Outputs

The time required to write the output status of built-in outputs is 41.1  $\mu$ s. Don't confuse this with the I/O response time that was discussed earlier.



### 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)		DL06
Nothing Connected	Min. & Max	0µs
Port 1	Send Min. / Max.	5.8/11.8 µs
	Rec. Min. / Max.	12.5/25.2 µs
Port 2	Send Min. / Max.	6.2/14.3 µs
	Rec. Min. / Max.	14.2/31.9 µs
LCD	Min. / Max.	4.8/49.2 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	DL06
Minimum	9 µs
Run Mode Max.	412 µs
Program Mode Max.	2.5 second

### 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.

### 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		DL06
Program Mode	Minimum	12.0µs
	Maximum	12.0µs
Run Mode	Minimum	20.0µs
	Maximum	27.0µs

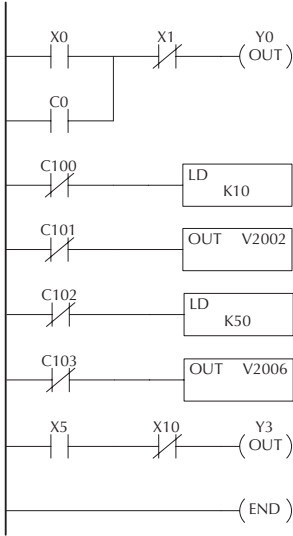
## Application Program Execution

The CPU processes the program from 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.

Just add the execution times for all the instructions in your program to determine to total execution time. Appendix C provides a complete list of the instruction execution times for the DL06 Micro PLC. For example, the execution time for running the program shown below is calculated as follows:

Instruction	Time
STR X0	.67 µs
OR C0	.51 µs
ANDN X1	.51 µs
OUT Y0	1.82 µs
STRN C100	.67 µs
LD K10	9.00 µs
STRN C101	.67 µs
OUT V2002	9.3 µs
STRN C102	.67 µs
LD K50	9.00 µs
STRN C103	.67 µs
OUT V2006	1.82 µs
STR X5	.67 µs
ANDN X10	.51 µs
OUT Y3	1.82 µs
END	12.80 µs
SUBT OTAL	51.11 µs

Overhead	DL06
Minimum	746.2 µs
Maximum	4352.4 µs



**TOTAL TIME = (Program execution time + Overhead) x 1.18**

The program above takes only 51.11 µs to execute during each scan. The DL06 spends 0.18 ms, on internal timed interrupt management, for every 1ms of instruction time. The total scan time is calculated by adding the program execution time to the overhead (shown above)and multiplying the result (ms) by 1.18. “Overhead” includes all other housekeeping and diagnostic tasks. The scan time will vary slightly from one scan to the next, because of fluctuation in overhead tasks.

**Program Control Instructions** — the DL06 CPUs offer additional instructions that can change the way the program executes. These instructions include FOR/NEXT loops, Subroutines, and Interrupt Routines. These instructions can interrupt the normal program flow and affect the program execution time. Chapter 5 provides detailed information on how these different types of instructions operate.

PLC Numbering Systems

If you are a new PLC user or are using our 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 our PLCs. The information you learn here applies to all of our PLCs!

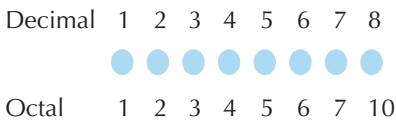
octal	BCD	?	binary
? 1482	? 3	0402	?
3A9	7	-961428	ASCII
1001011011			hexadecimal
	177	?	1011
decimal	A	72B	?
-300124			

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.

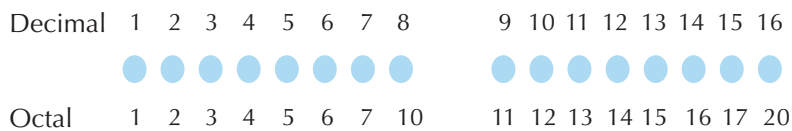
PLC Resources

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).

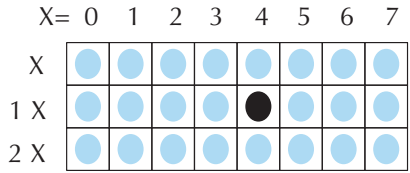


In the figure below, we have two groups of eight circles. Counting in octal we have “20” items, meaning 2 groups of eight, plus 0 individuals. Don't say “twenty”, say “two-zero octal”. This makes a clear distinction between number systems.



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

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

V-memory data is 16-bit binary, but we rarely program the data registers one bit at a time. We use instructions or viewing tools that let us work with 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 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. 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 can represent numbers from 0 to 65535. In storing BCD numbers, the range is reduced to only 0 to 9999. Many math instructions use Binary-Coded Decimal (BCD) data, and *Direct*SOFT32 and the handheld programmer allow us to enter and view data in BCD.

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.

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

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

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

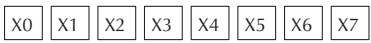
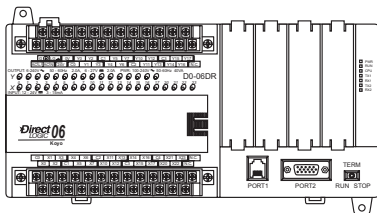


# Memory Map

With any PLC system, you generally have many different types of information to process. This includes input device status, output device status, various timing elements, 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 DL06 Micro PLCs. A memory map overview for the CPU follows the memory descriptions.

## Octal Numbering System

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

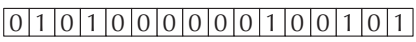


Discrete – On or Off, 1 bit

X0



Word Locations – 16 bits



## Discrete and Word Locations

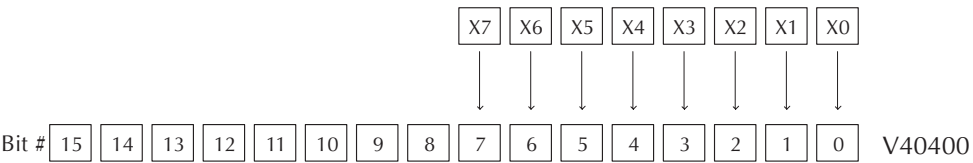
As you examine the different memory types, you'll notice two types of memory in the DL06, discrete and word memory. Discrete memory is one bit that can be either a 1 or a 0. Word memory is referred to as V memory (variable) and is a 16-bit location normally used to manipulate data/numbers, store data/numbers, etc.

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

## V Memory Locations for Discrete Memory Areas

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

8 Discrete (X) Input Points



These discrete memory areas and their corresponding V memory ranges are listed in the memory area table for DL06 Micro PLCs on the following pages.

### Input Points (X Data Type)

The discrete input points are noted by an X data type. There are 8 discrete input points and 256 discrete input addresses available with DL06 CPUs. In this example, the output point Y0 will be turned on when input X0 energizes.



### Output Points (Y Data Type)

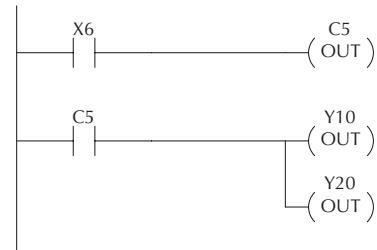
The discrete output points are noted by a Y data type. There are 6 discrete outputs and 256 discrete output addresses available with DL06 CPUs. In this example, output point Y1 will be turned on when input X1 energizes.



### Control Relays (C Data Type)

Control relays are discrete bits normally used to control the user program. The control relays do not represent a real world device, that is, they cannot be physically tied to switches, output coils, etc. They are internal to the CPU. Because of this, control relays can be programmed as discrete inputs or discrete outputs. These locations are used in programming the discrete memory locations (C) or the corresponding word location which contains 16 consecutive discrete locations.

In this example, memory location C5 will energize when input X6 turns on. The second rung shows a simple example of how to use a control relay as an input.

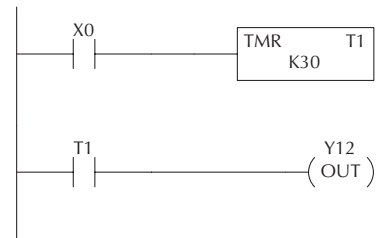


### Timers and Timer Status Bits (T Data Type)

Timer status bits 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.

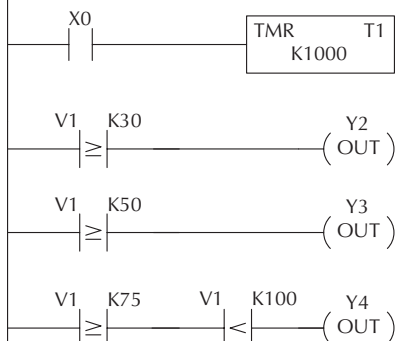
When the timer reaches the preset of 3 seconds (K of 30) timer status contact T1 turns on. When T1 turns on, output Y12 turns on. Turning off X0 resets the timer.



## Timer Current Values (V Data Type)

As mentioned earlier, some information is automatically stored in V memory. This is true for 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 can also be designated as TA0 (Timer Accumulated) for Timer 0, and TA1 for Timer 1.

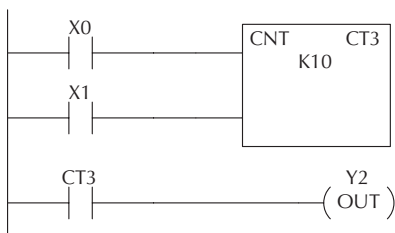
The primary reason for this is programming flexibility. The example shows how you can use relational contacts to monitor several time intervals from a single timer.



## Counters and Counter Status Bits (CT Data type)

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

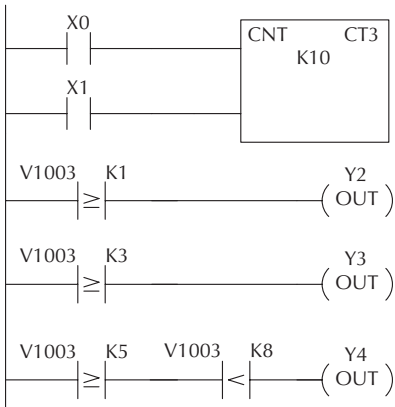
Each time contact X0 transitions from off to on, the counter increments by one. (If X1 comes on, the counter is reset to zero.) When the counter reaches the preset of 10 counts (K of 10) counter status contact CT3 turns on. When CT3 turns on, output Y2 turns on.



## Counter Current Values (V Data Type)

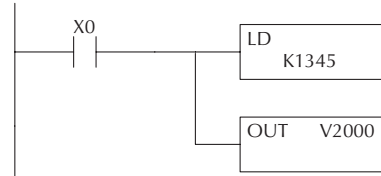
Just like the timers, the counter current values are also automatically stored in V memory. For example, V1000 holds the current value for Counter CT0, V1001 holds the current value for Counter CT1, etc. These can also be designated as CTA0 (Counter Accumulated) for Counter 0 and CTA01 for Timer 1.

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



## Word Memory (V Data Type)

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

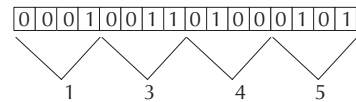


## Stages (S Data type)

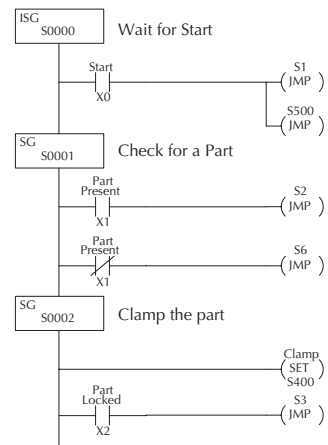
Stages are used in RLL<sup>PLUS</sup> programs to create a structured program, similar to a flowchart. Each program Stage denotes a program segment. When the program segment, or Stage, is active, the logic within that segment is executed. If the Stage is off, or inactive, the logic is not executed and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

Word Locations – 16 bits



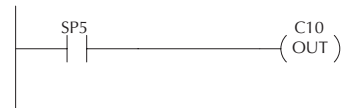
Ladder Representation



## 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.



SP4: 1 second clock  
SP5: 100 ms clock  
SP6: 50 ms clock

## DL06 System V-memory

### System Parameters and Default Data Locations (V Data Type)

The DL06 PLCs reserve several V-memory locations for storing system parameters or certain types of system data. These memory locations store things like the error codes, High-Speed I/O data, and other types of system setup information.

System V-memory	Description of Contents	Default Values / Ranges
V700-V707	Sets the V-memory location for option card in slot 1	N/A
V710-V717	Sets the V-memory location for option card in slot 2	N/A
V720-V727	Sets the V-memory location for option card in slot 3	N/A
V730-V737	Sets the V-memory location for option card in slot 4	N/A
V3630-V3707	The default location for multiple preset values for UP/DWN and UP Counter 1 or pulse catch function	N/A
V3710-V3767	The default location for multiple preset values for UP/DWN and UP Counter 2	N/A
V7620-V7627	"Locations for DV-1000 operator interface parameters. Sets the V-memory location that contains the value. Sets the V-memory location that contains the message. Sets the total number (1 - 32) of V-memory locations to be displayed. Sets the V-memory location that contains the numbers to be displayed. Sets the V-memory location that contains the character code to be displayed. Contains the function number that can be assigned to each key. Powerup operational mode. Change preset value."	V0 - V3760 V0 - V37601 - 32 V0 - V3760 V0 - V3760 V-memory for X, Y, or C0, 1, 2, 3, 12 Default = 0000
V7630	Starting location for the multi-step presets for channel 1. The default value is 3630, which indicates the first value should be obtained from V3630. Since there are 24 presets available, the default range is V3630 - V3707. You can change the starting point if necessary.	Default: V3630 Range: V0- V3710
V7631	Starting location for the multi-step presets for channel 2. The default value is 3710, which indicates the first value should be obtained from V3710. Since there are 24 presets available, the default range is V3710 - V3767. You can change the starting point if necessary.	Default: V3710 Range: V0- V3710
V7632	Setup Register for Pulse Output	N/A
V7633	Sets the desired function code for the high speed counter, interrupt, pulse catch, pulse train, and input filter. Location can also be used to set the power-up in Run Mode option.	Default: 0060 Lower Byte Range: Range: 10 - Counter 20 - Quadrature 30 - Pulse Out 40 - Interrupt 50 - Pulse Catch 60 - Filtered discrete In. Upper Byte Range: Bits 8-11, 14, 15: Unused, Bit 13: Power-up in RUN, only if Mode Switch is in TERM position. Bit 12 is used to enable the battery.
V7634	X0 Setup Register for High-Speed I/O functions	Default: 1006
V7635	X1 Setup Register for High-Speed I/O functions	Default: 1006
V7636	X2 Setup Register for High-Speed I/O functions	Default: 1006
V7637	X3 Setup Register for High-Speed I/O functions	Default: 1006
V7640	PID Loop table beginning address	"V1200 - V3737 V10000 - V17777"

## Chapter 4: CPU Specifications and Operation

System V-memory	Description of Contents	Default Values / Ranges
V7641	Number of Loops	1-8
V7642	Error Code - V-memory Error location for Loop Table	
V7643-V7647	Reserved	
V7650	Port 2: Setup for V-memory address for Non-procedure protocol	V1200 – V7377 V10000 - V17777
V7653	Port 2: Setup for terminate code for Non-procedure protocol	
V7655	Port 2: Setup for the protocol, time-out, and the response delay time.	
V7656	Port 2: Setup for the station number, baud rate, STOP bit, and parity.	
V7657	Port 2: Setup completion code used to notify the completion of the parameter setup	
V7660	Scan control setup: Keeps the scan control mode.	
V7661	Setup timer over counter: Counts the times the actual scan time exceeds the user setup time.	
V7662-V7717	Reserved	
V7720-V7722	Locations for DV-1000 operator interface parameters.	
V7720	Titled Timer preset value pointer	
V7721	Title Counter preset value pointer	
V7722	HiByte-Titled Timer preset block size, LoByte-Titled Counter preset block size	
V7723-V7737	Reserved	
V7740	Port 1 and Port 2: Communication Auto Reset Timer Setup	Default: 3030
V7741-V7746	Reserved	
V7747	Location contains a 10mS counter (0-99). This location increments once every 10 mS	
V7750	Reserved	
V7751	Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed.	
V7752	I/O Configuration Error: Current ID code of error slot	
V7753	I/O Configuration Error: Old ID code of error slot	
V7754	I/O Configuration Error: error slot number	
V7755	Error code — stores the fatal error code.	
V7756	Error code — stores the major error code.	
V7757	Error code — stores the minor error code.	
V7760-V7762	Reserved	
V7763	Program address where syntax error exists	
V7764	Syntax error code	
V7765	Scan counter — stores the total number of scan cycles that have occurred since the last Program Mode to Run Mode transition.	
V7766	Contains the number of seconds on the clock (00-59)	
V7767	Contains the number of minutes on the clock (00-59)	
V7770	Contains the number of hours on the clock (00-23)	
V7771	Contains the day of the week (Mon., Tues., Wed., etc.)	
V7772	Contains the day of the month (01, 02, etc.)	
V7773	Contains the month (01 to 12)	
V7774	Contains the year (00 to 99)	
V7775	Scan — stores the current scan time (milliseconds).	
V7776	Scan — stores the minimum scan time that has occurred since the last	
V7777	Program Mode to Run Mode transition (milliseconds).	
V37700-V37737	For remote I/O	

## DL06 Memory Map

Memory Type	Discrete Memory Reference (octal)	Word Memory Reference (octal)	Decimal	Symbol
Input Points	X0 – X777	V40400 - V40437	512	X0 ┌─┴─┐
Output Points	Y0 – Y777	V40500 – V40537	512	Y0 ┌─( )─┐
Control Relays	C0 – C1777	V40600 - V40677	1024	C0      C0 ┌─┴─┐ ┌─( )─┐
Special Relays	SP0 – SP777	V41200 – V41237	512	SP0 ┌─┴─┐
Timers	T0 – T377	V41100 – V41117	256	┌─┐ TMR    T0 └─┘      K100
Timer Current Values	None	V0 – V377	256	V0    K100 ┌─┴─┐
Timer Status Bits	T0 – T377	V41100 – V41117	256	T0 ┌─┴─┐
Counters	CT0 – CT177	V41140 – V41147	128	┌─┐ CNT    CT0 └─┘      K10
Counter Current Values	None	V1000 – V1177	128	V1000    K100 ┌─┴─┐
Counter Status Bits	CT0 – CT177	V41140 – V41147	128	CT0 ┌─┴─┐
Data Words	None	V400-V677 V1200 – V7377 V10000 - V17777	192 3200 4096	None specific, used with many instructions
Data Words Non-volatile	None	V7400 – V7577	128	None specific, used with many instructions
Stages	S0 – S1777	V41000 – V41017	1024	┌─┐ SG      S0 └─┘    S001    ┌─┴─┐
Remote I/O	GX0-GX3777 GY0-GY3777	V40000-V40177 V40200-V40377	2048 2048	GX0      GY0 ┌─┴─┐    ┌─( )─┐
System parameters	None	V700-V777 V7600 – V7777 V36000-V37777	64 128 1024	None specific, used for various purposes

1-The DL06 systems are limited to 20 discrete inputs and 16 discrete outputs with the present available hardware, but 512 point addresses exist.

### 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 DL06's twenty integrated physical inputs and 16 integrated physical outputs in addition to up to 64 inputs and 64 outputs for option cards. Actual available references are X0 to X777 (V40400 – V40437) and Y0 to Y777 (V40500 – V40537).

MSB		DL06 Input (X) and Output (Y) Points														LSB		X Input Address	Y Output Address
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0				
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000			V40400	V40500
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020			V40401	V40501
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040			V40402	V40502
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060			V40403	V40503
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100			V40404	V40504
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120			V40405	V40505
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140			V40406	V40506
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160			V40407	V40507
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200			V40410	V40510
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220			V40411	V40511
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240			V40412	V40512
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260			V40413	V40513
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300			V40414	V40514
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320			V40415	V40515
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340			V40416	V40516
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360			V40417	V40517
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400			V40420	V40520
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420			V40421	V40521
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440			V40422	V40522
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460			V40423	V40523
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500			V40424	V40524
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520			V40425	V40525
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540			V40426	V40526
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560			V40427	V40527
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600			V40430	V40530
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620			V40431	V40531
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640			V40432	V40532
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660			V40433	V40533
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700			V40434	V40534
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720			V40435	V40535
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740			V40436	V40536
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760			V40437	V40537



### Stage Control / Status Bit Map

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

MSB	DL06 Stage (S) Control Bits															LSB	Address
17	16	15	14	13	12	11	10	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	

## Chapter 4: CPU Specifications and Operation

MSB	DL06 Stage (S) Control Bits															LSB	Address
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0		
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500	V41024	
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520	V41025	
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540	V41026	
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560	V41027	
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600	V41030	
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620	V41031	
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640	V41032	
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660	V41033	
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700	V41034	
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720	V41035	
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740	V41036	
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760	V41037	
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	

## Control Relay Bit Map

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

MSB	DL06 Control Relays (C)															LSB	Address
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0		
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000		V40600
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020		V40601
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040		V40602
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060		V40603
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100		V40604
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120		V40605
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140		V40606
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160		V40607
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200		V40610
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220		V40611
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240		V40612
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260		V40613
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300		V40614
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320		V40615
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340		V40616
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360		V40617
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400		V40620
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420		V40621
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440		V40622
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460		V40623
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500		V40624
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520		V40625
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540		V40626
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560		V40627
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600		V40630
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620		V40631
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640		V40632
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660		V40633
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700		V40634
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720		V40635
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740		V40636
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760		V40637

## Chapter 4: CPU Specifications and Operation

MSB	DL06 Control Relays (C)														LSB	Address
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0	
1017	1016	1015	1014	1013	1012	1011	1010	1007	1006	1005	1004	1003	1002	1001	1000	V40640
1037	1036	1035	1034	1033	1032	1031	1030	1027	1026	1025	1024	1023	1022	1021	1020	V40641
1057	1056	1055	1054	1053	1052	1051	1050	1047	1046	1045	1044	1043	1042	1041	1040	V40642
1077	1076	1075	1074	1073	1072	1071	1070	1067	1066	1065	1064	1063	1062	1061	1060	V40643
1117	1116	1115	1114	1113	1112	1111	1110	1107	1106	1105	1104	1103	1102	1101	1100	V40644
1137	1136	1135	1134	1133	1132	1131	1130	1127	1126	1125	1124	1123	1122	1121	1120	V40645
1157	1156	1155	1154	1153	1152	1151	1150	1147	1146	1145	1144	1143	1142	1141	1140	V40646
1177	1176	1175	1174	1173	1172	1171	1170	1167	1166	1165	1164	1163	1162	1161	1160	V40647
1217	1216	1215	1214	1213	1212	1211	1210	1207	1206	1205	1204	1203	1202	1201	1200	V40650
1237	1236	1235	1234	1233	1232	1231	1230	1227	1226	1225	1224	1223	1222	1221	1220	V40651
1257	1256	1255	1254	1253	1252	1251	1250	1247	1246	1245	1244	1243	1242	1241	1240	V40652
1277	1276	1275	1274	1273	1272	1271	1270	1267	1266	1265	1264	1263	1262	1261	1260	V40653
1317	1316	1315	1314	1313	1312	1311	1310	1307	1306	1305	1304	1303	1302	1301	1300	V40654
1337	1336	1335	1334	1333	1332	1331	1330	1327	1326	1325	1324	1323	1322	1321	1320	V40655
1357	1356	1355	1354	1353	1352	1351	1350	1347	1346	1345	1344	1343	1342	1341	1340	V40656
1377	1376	1375	1374	1373	1372	1371	1370	1367	1366	1365	1364	1363	1362	1361	1360	V40657
1417	1416	1415	1414	1413	1412	1411	1410	1407	1406	1405	1404	1403	1402	1401	1400	V40660
1437	1436	1435	1434	1433	1432	1431	1430	1427	1426	1425	1424	1423	1422	1421	1420	V40661
1457	1456	1455	1454	1453	1452	1451	1450	1447	1446	1445	1444	1443	1442	1441	1440	V40662
1477	1476	1475	1474	1473	1472	1471	1470	1467	1466	1465	1464	1463	1462	1461	1460	V40663
1517	1516	1515	1514	1513	1512	1511	1510	1507	1506	1505	1504	1503	1502	1501	1500	V40664
1537	1536	1535	1534	1533	1532	1531	1530	1527	1526	1525	1524	1523	1522	1521	1520	V40665
1557	1556	1555	1554	1553	1552	1551	1550	1547	1546	1545	1544	1543	1542	1541	1540	V40666
1577	1576	1575	1574	1573	1572	1571	1570	1567	1566	1565	1564	1563	1562	1561	1560	V40667
1617	1616	1615	1614	1613	1612	1611	1610	1607	1606	1605	1604	1603	1602	1601	1600	V40670
1637	1636	1635	1634	1633	1632	1631	1630	1627	1626	1625	1624	1623	1622	1621	1620	V40671
1657	1656	1655	1654	1653	1652	1651	1650	1647	1646	1645	1644	1643	1642	1641	1640	V40672
1677	1676	1675	1674	1673	1672	1671	1670	1667	1666	1665	1664	1663	1662	1661	1660	V40673
1717	1716	1715	1714	1713	1712	1711	1710	1707	1706	1705	1704	1703	1702	1701	1700	V40674
1737	1736	1735	1734	1733	1732	1731	1730	1727	1726	1725	1724	1723	1722	1721	1720	V40675
1757	1756	1755	1754	1753	1752	1751	1750	1747	1746	1745	1744	1743	1742	1741	1740	V40676
1777	1776	1775	1774	1773	1772	1771	1770	1767	1766	1765	1764	1763	1762	1761	1760	V40677

## Timer Status Bit Map

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

MSB	DL06 Timer (T) Contacts															LSB	Address
17	16	15	14	13	12	11	10	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
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020		V41101
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040		V41102
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060		V41103
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100		V41104
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120		V41105
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140		V41106
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160		V41107
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

## Counter Status Bit Map

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

MSB	DL06 Counter (CT) Contacts															LSB	Address
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0		
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000		V41140
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020		V41141
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040		V41142
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060		V41143
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100		V41144
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120		V41145
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140		V41146
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160		V41147

## Remote I/O Bit Map

This table provides a listing of the individual remote I/O points associated with each V-memory address bit.

MSB	Remote I/O (GX) and (GY) Points															LSB	GX Address	GY Address
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0			
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000		V40000	V40200
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020		V40001	V40201
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040		V40002	V40202
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060		V40003	V40203
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100		V40004	V40204
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120		V40005	V40205
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140		V40006	V40206
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160		V40007	V40207
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200		V40010	V40210
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220		V40011	V40211
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240		V40012	V40212
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260		V40013	V40213
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300		V40004	V40214
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320		V40015	V40215
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340		V40016	V40216
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360		V40007	V40217
417	416	415	414	413	412	411	410	407	406	405	404	403	402	401	400		V40020	V40220
437	436	435	434	433	432	431	430	427	426	425	424	423	422	421	420		V40021	V40221
457	456	455	454	453	452	451	450	447	446	445	444	443	442	441	440		V40022	V40222
477	476	475	474	473	472	471	470	467	466	465	464	463	462	461	460		V40023	V40223
517	516	515	514	513	512	511	510	507	506	505	504	503	502	501	500		V40024	V40224
537	536	535	534	533	532	531	530	527	526	525	524	523	522	521	520		V40025	V40225
557	556	555	554	553	552	551	550	547	546	545	544	543	542	541	540		V40026	V40226
577	576	575	574	573	572	571	570	567	566	565	564	563	562	561	560		V40027	V40227
617	616	615	614	613	612	611	610	607	606	605	604	603	602	601	600		V40030	V40230
637	636	635	634	633	632	631	630	627	626	625	624	623	622	621	620		V40031	V40231
657	656	655	654	653	652	651	650	647	646	645	644	643	642	641	640		V40032	V40232
677	676	675	674	673	672	671	670	667	666	665	664	663	662	661	660		V40033	V40233
717	716	715	714	713	712	711	710	707	706	705	704	703	702	701	700		V40034	V40234
737	736	735	734	733	732	731	730	727	726	725	724	723	722	721	720		V40035	V40235
757	756	755	754	753	752	751	750	747	746	745	744	743	742	741	740		V40036	V40236
777	776	775	774	773	772	771	770	767	766	765	764	763	762	761	760		V40037	V40237

MSB	DL06 Remote I/O (GX) and (GY) Points															LSB	GX	GY
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0		Address	Address
1017	1016	1015	1014	1013	1012	1011	1010	1007	1006	1005	1004	1003	1002	1001	1000		V40040	V40240
1037	1036	1035	1034	1033	1032	1031	1030	1027	1026	1025	1024	1023	1022	1021	1020		V40041	V40241
1057	1056	1055	1054	1053	1052	1051	1050	1047	1046	1045	1044	1043	1042	1041	1040		V40042	V40242
1077	1076	1075	1074	1073	1072	1071	1070	1067	1066	1065	1064	1063	1062	1061	1060		V40043	V40243
1117	1116	1115	1114	1113	1112	1111	1110	1107	1106	1105	1104	1103	1102	1101	1100		V40044	V40244
1137	1136	1135	1134	1133	1132	1131	1130	1127	1126	1125	1124	1123	1122	1121	1120		V40045	V40245
1157	1156	1155	1154	1153	1152	1151	1150	1147	1146	1145	1144	1143	1142	1141	1140		V40046	V40246
1177	1176	1175	1174	1173	1172	1171	1170	1167	1166	1165	1164	1163	1162	1161	1160		V40047	V40247
1217	1216	1215	1214	1213	1212	1211	1210	1207	1206	1205	1204	1203	1202	1201	1200		V40050	V40250
1237	1236	1235	1234	1233	1232	1231	1230	1227	1226	1225	1224	1223	1222	1221	1220		V40051	V40251
1257	1256	1255	1254	1253	1252	1251	1250	1247	1246	1245	1244	1243	1242	1241	1240		V40052	V40252
1277	1276	1275	1274	1273	1272	1271	1270	1267	1266	1265	1264	1263	1262	1261	1260		V40053	V40253
1317	1316	1315	1314	1313	1312	1311	1310	1307	1306	1305	1304	1303	1302	1301	1300		V40054	V40254
1337	1336	1335	1334	1333	1332	1331	1330	1327	1326	1325	1324	1323	1322	1321	1320		V40055	V40255
1357	1356	1355	1354	1353	1352	1351	1350	1347	1346	1345	1344	1343	1342	1341	1340		V40056	V40256
1377	1376	1375	1374	1373	1372	1371	1370	1367	1366	1365	1364	1363	1362	1361	1360		V40057	V40257
1417	1416	1415	1414	1413	1412	1411	1410	1407	1406	1405	1404	1403	1402	1401	1400		V40060	V40260
1437	1436	1435	1434	1433	1432	1431	1430	1427	1426	1425	1424	1423	1422	1421	1420		V40061	V40261
1457	1456	1455	1454	1453	1452	1451	1450	1447	1446	1445	1444	1443	1442	1441	1440		V40062	V40262
1477	1476	1475	1474	1473	1472	1471	1470	1467	1466	1465	1464	1463	1462	1461	1460		V40063	V40263
1517	1516	1515	1514	1513	1512	1511	1510	1507	1506	1505	1504	1503	1502	1501	1500		V40064	V40264
1537	1536	1535	1534	1533	1532	1531	1530	1527	1526	1525	1524	1523	1522	1521	1520		V40065	V40265
1557	1556	1555	1554	1553	1552	1551	1550	1547	1546	1545	1544	1543	1542	1541	1540		V40066	V40266
1577	1576	1575	1574	1573	1572	1571	1570	1567	1566	1565	1564	1563	1562	1561	1560		V40067	V40267
1617	1616	1615	1614	1613	1612	1611	1610	1607	1606	1605	1604	1603	1602	1601	1600		V40070	V40270
1637	1636	1635	1634	1633	1632	1631	1630	1627	1626	1625	1624	1623	1622	1621	1620		V40071	V40271
1657	1656	1655	1654	1653	1652	1651	1650	1647	1646	1645	1644	1643	1642	1641	1640		V40072	V40272
1677	1676	1675	1674	1673	1672	1671	1670	1667	1666	1665	1664	1663	1662	1661	1660		V40073	V40273
1717	1716	1715	1714	1713	1712	1711	1710	1707	1706	1705	1704	1703	1702	1701	1700		V40074	V40274
1737	1736	1735	1734	1733	1732	1731	1730	1727	1726	1725	1724	1723	1722	1721	1720		V40075	V40275
1757	1756	1755	1754	1753	1752	1751	1750	1747	1746	1745	1744	1743	1742	1741	1740		V40076	V40276
1777	1776	1775	1774	1773	1772	1771	1770	1767	1766	1765	1764	1763	1762	1761	1760		V40077	V40277

MSB	DL06 Remote I/O (GX) and (GY) Points															LSB	GX Address	GY Address
17	16	15	14	13	12	11	10	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	2736	2735	2734	2733	2732	2731	2730		V40136	V40336
2777	2776	2775	2774	2773	2772	2771	2770	2767	2766	2765	2764	2763	2762	2761	2760		V40137	V40337

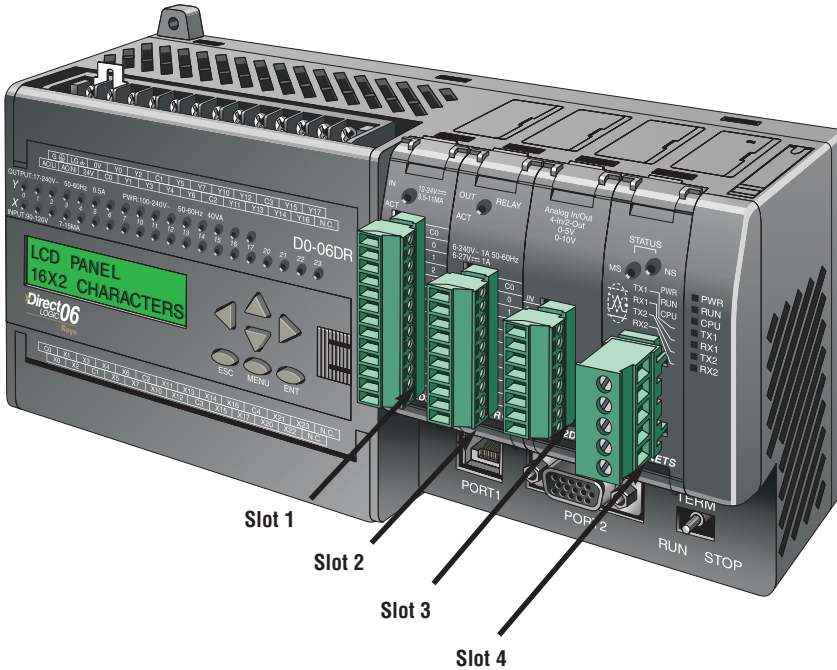


MSB	DL06 Remote I/O (GX) and (GY) Points															LSB	GX Address	GY Address
17	16	15	14	13	12	11	10	7	6	5	4	3	2	1	0			
3017	3016	3015	3014	3013	3012	3011	3010	3007	3006	3005	3004	3003	3002	3001	3000		V40140	V40340
3037	3036	3035	3034	3033	3032	3031	3030	3027	3026	3025	3024	3023	3022	3021	3020		V40141	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

# Module Placement

## Slot Numbering

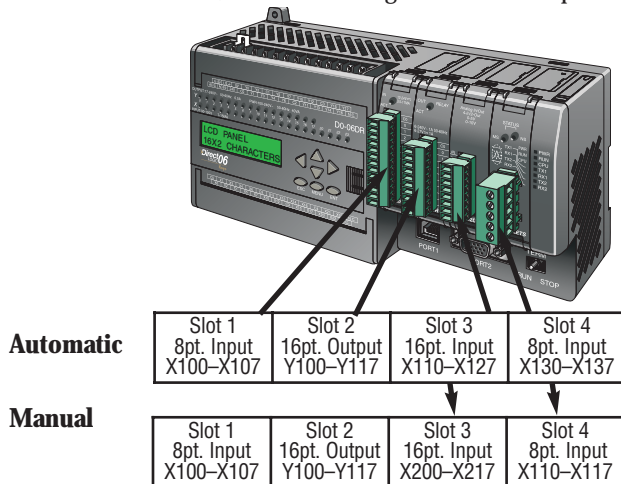
The DL06 has four slots, which are numbered as follows:



## Automatic I/O Configuration

The DL06 CPUs automatically detect any installed I/O modules (including specialty modules) at powerup, and establish the correct I/O configuration and addresses. This applies to modules located in local and expansion I/O bases. For most applications, you will never have to change the configuration.

I/O addresses use octal numbering, starting at X100 and Y100 in the slot next to the CPU. The addresses are assigned in groups of 8, or 16 depending on the number of points for the I/O module. The discrete input and output modules can be mixed in any order, but there may be restrictions placed on some specialty modules. The following diagram shows the I/O numbering convention for an example system. Both the Handheld Programmer and DirectSOFT32 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 *DirectSOFT32*, the PLC Configure I/O menu option would be used.



## Manual I/O Configuration

It may never become necessary, but DL06 CPUs allow manual I/O address assignments for any I/O slot(s). You can manually modify an auto configuration to match arbitrary I/O numbering. For example, two adjacent input modules can have starting addresses at X100 and X200. Use DirectSOFT32 PLC Configure I/O menu option to assign manual I/O address. In automatic configuration, the addresses are assigned on 8-point boundaries. Manual configuration, however, assumes that all modules are at least 16 points, so you can only assign addresses that are a multiple of 20 (octal). For example, X130 and Y150 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 DL06 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.



### Power Budgeting

The DL06 has four option card slots. To determine whether the combination of cards you select will have sufficient power, you will need to perform a power budget calculation.

#### Power supplied

Power is supplied from two sources, the internal base unit power supply and, if required, an external supply (customer furnished). The D0-06xx (AC powered) PLCs supply a limited amount of 24VDC power. The 24VDC output can be used to power external devices. For power budgeting, start by considering the power supplied by the base unit. All DL06 PLCs supply the same amount of 5VDC power. Only the AC units offer 24VDC auxiliary power. Be aware of the trade-off between 5VDC power and 24VDC power. The amount of 5VDC power available depends on the amount of 24VDC power being used, and the amount of 24VDC power available depends on the amount of 5VDC power consumed. Determine the amount of internally supplied power from the table on the following page.

#### Power required by base unit

Because of the different I/O configurations available in the DL06 family, the power consumed by the base unit itself varies from model to model. Subtract the amount of power required by the base unit from the amount of power supplied by the base unit. Be sure to subtract 5VDC and 24VDC amounts.

#### Power required by option cards

Next, subtract the amount of power required by the option cards you are planning to use. Again, remember to subtract both 5VDC and 24VDC. If your power budget analysis shows surplus power available, you should have a workable configuration.

## DL06 Power Supplied by Base Units

Part Number	5 VDC (mA)	24 VDC (mA)
D0-06xx	<1500mA	300mA
	<2000mA	200mA
D0-06xx-D	1500mA	none

If the 5VDC loading is less than 2000mA but more than 1500mA, than available 24VDC supply current is 200mA.

If the 5VDC loading is less than 1500mA, then the available 24VDC current is 300mA.

## DL06 Base Unit Power Required

Part Number	5 VDC (mA)	24 VDC (mA)
D0-06AA	800mA	none
D0-06AR	900mA	none
D0-06DA	800mA	none
D0-06DD1	600mA	280mA*
D0-06DD2	600mA	none
D0-06DR	950mA	none
D0-06DD1-D	600mA	none
D0-06DR-D	950mA	none

## DL06 Power Consumed by Other Devices

Part Number	5 VDC (mA)	24 VDC (mA)
D0-06LCD	50mA	none
D2-HPP	200mA	none
DV1000	150mA	none

## DL06 Power Consumed by Option Cards

Part number	5 VDC	24 VDC
D0-07CDR	130mA	none
D0-08CDD1	100mA	none
D0-08TR	280mA	none
D0-10ND3	35mA	none
D0-10TD1	150mA	none
D0-10TD2	150mA	none
D0-16ND3	35mA	none
D0-16TD1	200mA	none
D0-16TD2	200mA	none
F0-04AD-1	50mA	none
F0-2AD2DA-2	50mA	30mA
F0-4AD2DA-1	100mA	40mA
F0-4AD2DA-2	100mA	none
D0-DEVNETS	45mA	none

## Power Budgeting Example

Power Source		5VDC power (mA)	24VDC power (mA)
D0-06DD1 (select row A or row B)	A	1500mA	300mA
	B	2000mA	200mA
Current Required		5VDC power (mA)	24VDC power (mA)
D0-06DD1		600mA	280mA*
D0-16ND3		35mA	0
D0-10TD1		150mA	0
D0-08TR		280mA	0
F0-4AD2DA-2		100mA	0
D0-06LCD		50mA	0
Total Used		1215mA	280mA
Remaining	A	285mA	20mA
	B	785mA	note 1

\* Auxiliary 24VDC used to power V+ terminal of D0-06DD1 sinking outputs.



Note 1: If the PLC's auxiliary 24VDC power source is used to power the sinking outputs, use power choice A, above.

## Configuring the DL06's Comm Ports

This section describes how to configure the CPU's built-in networking ports, for either MODBUS or DirectNET. This will allow you to connect the DL06 PLC system directly to MODBUS networks using the RTU protocol, or to other devices on a DirectNET 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 DirectNET, order our DirectNET manual, part number DA-DNET-M.



*Note: For information about the MODBUS protocol see the Group Schneider Web site at: [www.schneiderautomation.com](http://www.schneiderautomation.com). At the main menu, select Support/Services, Modbus, Modbus Technical Manuals, P1-MBUS-300 Modbus Protocol Reference Guide or search for PIMBUS300. For more information about the **DirectNET** protocol, order our **DirectNET** user manual, part number DA-DNET-M,*

### DL06 Port Specifications

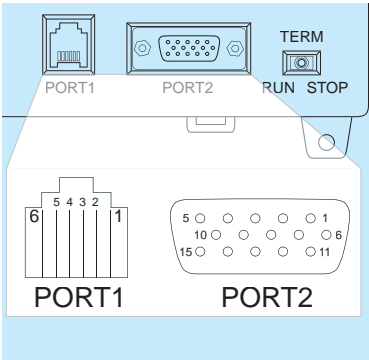
#### Communications Port 1

**Port 1** Connects to HPP, DirectSOFT32, operator interfaces, etc.  
6-pin, RS232C  
Communication speed (baud): 9600 (fixed)  
Parity: odd (fixed)  
Station Address: 1 (fixed)  
8 data bits  
1 start, 1 stop bit  
Asynchronous, half-duplex, DTE  
Protocol (auto-select): K-sequence (slave only), DirectNET (slave only), MODBUS (slave only)

#### Communications Port 2

**Port 2** Connects to HPP, DirectSOFT32, operator interfaces, etc.  
15-pin, multifunction port, RS232C, RS422, RS485  
Communication speed (baud): 300, 600, 1200, 2400, 4800, 9600, 19200, 38400  
Parity: odd (default), even, none  
Station Address: 1 (default)  
8 data bits  
1 start, 1 stop bit  
Asynchronous, half-duplex, DTE  
Protocol (auto-select): K-sequence (slave only), DirectNET (master/slave), MODBUS (master/slave), non-sequence/print/ASCII in/out

### DL06 Port Pinouts



#### Port 1 Pin Descriptions

1	0V	Power (-) connection (GND)
2	5V	Power (+) connection
3	RXD	Receive data (RS-232C)
4	TXD	Transmit data (RS-232C)
5	5V	Power (+) connection
6	0V	Power (-) connection (GND)

#### Port 2 Pin Descriptions

1	5V	Power (+) connection
2	TXD	Transmit data (RS-232C)
3	RXD	Receive data (RS-232C)
4	RTS	Ready to send
5	CTS	Clear to send
6	RXD-	Receive data (-) (RS-422/485)
7	0V	Power (-) connection (GND)
8	0V	Power (-) connection (GND)
9	TXD+	Transmit data (+) (RS-422/485)
10	TXD-	Transmit data (-) (RS-422/485)
11	RTS+	Ready to send (+) (RS-422/485)
12	RTS-	Ready to send (-) (RS-422/485)
13	RXD+	Receive data (+) (RS-422/485)
14	CTS+	Clear to send (+) (RS-422/485)
15	CTS-	Clear to send (-) (RS-422/485)

## Choosing a Network Specification

The DL06 PLC's multi-function port gives you the option of using RS-232C, RS-422, or RS-485 specifications. First, determine whether the network will be a 2-wire RS-232C type, a 4-wire RS-422 type, or a 2-wire/4-wire RS-485 type.

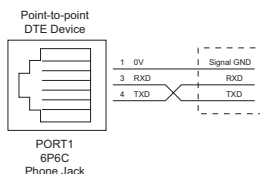
The RS-232C specification is simple to implement for networks of shorter distances (15 meters max) and where communication is only required between two devices. The RS-422 and RS-485 signals are for networks that cover longer distances (1000 meters max.) and for multi-drop networks (from 2 to 247 devices).



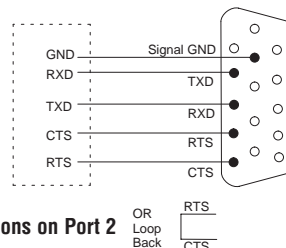
*Note: Termination resistors are required at both ends of RS-422 and RS-485 networks. It is necessary to select resistors that match the impedance rating of the cable (between 100 and 500 ohms).*

### RS-232 Network

Normally, the RS-232 signals are used for shorter distances (15 meters maximum), for communications between two devices.



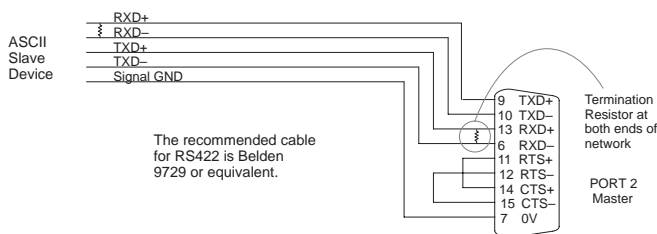
Connections on Port 1



Connections on Port 2

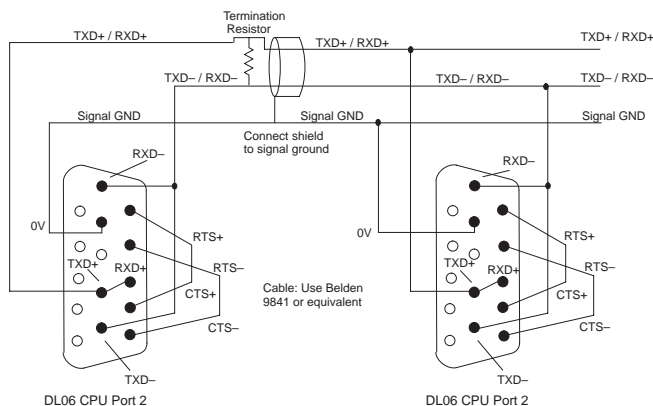
### RS-422 Network

RS-422 signals are for long distances (1000 meters maximum). Use terminator resistors at both ends of RS-422 network wiring, matching the impedance rating of the cable (between 100 and 500 ohms).



### RS-485 Network

RS-485 signals are for longer distances (1000 meters max) and for multi-drop networks. Use termination resistors at both ends of RS-485 network wiring, matching the impedance rating of the cable (between 100 and 500 ohms).

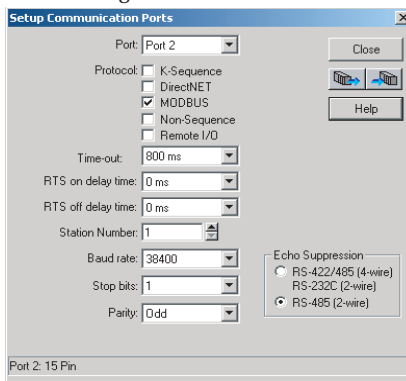


# Connecting to MODBUS and *DirectNET* Networks

## MODBUS Port Configuration

In *DirectSOFT32*, choose the PLC menu, then Setup, then “Secondary Comm Port”.

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



- **Timeout:** amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS ON / OFF Delay Time:** The RTS ON Delay Time specifies the time the DL06 waits to send the data after it has raised the RTS signal line. The RTS OFF Delay Time specifies the time the DL06 waits to release the RTS signal line after the data has been sent. *When using the DL06 on a multi-drop network, the RTS ON Delay time must be set to at least 5ms and the RTS OFF Delay time must be set to at least 2ms. If you encounter problems, the time can be increased.*
- **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 DL06 network instructions used in Master mode will access only slaves 1 to 99. Each slave must have a unique number. At powerup, the port is automatically a slave, unless and until the DL06 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, 1200, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value. Refer to the appropriate product manual for details.
- **Stop Bits:** Choose 1 or 2 stop bits for use in the protocol.
- **Parity:** Choose none, even, or odd parity for error checking.
- **Echo Suppression:** Select the appropriate radio button based on the wiring configuration used on port 2.



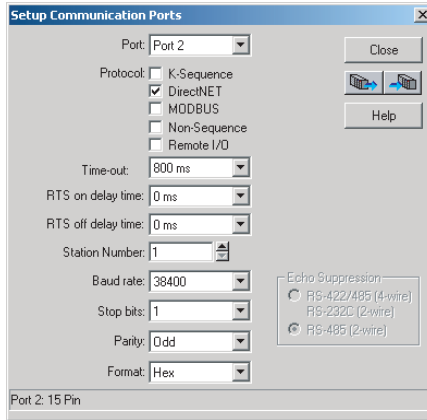
Then click the button indicated to send the Port configuration to the CPU, and click Close.



## DirectNET Port Configuration

In *DirectSOFT32*, choose the PLC menu, then Setup, then “Secondary Comm Port”.

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



- **Timeout:** Amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS ON / OFF Delay Time:** The RTS ON Delay Time specifies the time the DL06 waits to send the data after it has raised the RTS signal line. The RTS OFF Delay Time specifies the time the DL06 waits to release the RTS signal line after the data has been sent. When using the DL06 on a multi-drop network, the RTS ON Delay time must be set to at least 5ms and the RTS OFF Delay time must be set to at least 2ms. If you encounter problems, the time can be increased.
- **Station Number:** For making the CPU port a DirectNET master, choose “1”. The allowable range for DirectNET slaves is from 1 to 90 (each slave must have a unique number). At powerup, the port is automatically a slave, unless and until the DL06 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, 1200, 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.

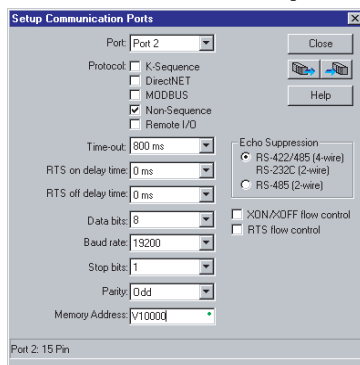
# Non-Sequence Protocol (ASCII In/Out and PRINT)

## MODBUS Port Configuration

Configuring port 2 on the DL06 for Non-Sequence allows the CPU to use port 2 to either read or write raw ASCII strings using the ASCII instructions. See the ASCII In/Out instructions and the PRINT instruction in chapter 5.

In *DirectSOFT32*, choose the PLC menu, then Setup, then “Secondary Comm Port”.

- **Port:** From the port number list box at the top, choose “Port 2”.
- **Protocol:** Click the check box to the left of “Non-Sequence”.



- **Timeout:** Amount of time the port will wait after it sends a message to get a response before logging an error.
- **RTS On Delay Time:** The amount of time between raising the RTS line and sending the data.
- **RTS Off Delay Time:** The amount of time between resetting the RTS line after sending the data.
- **Data Bits:** Select either 7-bits or 8-bits to match the number of data bits specified for the connected devices.
- **Baud Rate:** The available baud rates include 300, 600, 900, 2400, 4800, 9600, 19200, and 38400 baud. Choose a higher baud rate initially, reverting to lower baud rates if you experience data errors or noise problems on the network. Important: You must configure the baud rates of all devices on the network to the same value. Refer to the appropriate product manual for details.
- **Stop Bits:** Choose 1 or 2 stop bits to match the number of stop bits specified for the connected devices.
- **Parity:** Choose none, even, or odd parity for error checking. Be sure to match the parity specified for the connected devices.
- **Echo Suppression:** Select the appropriate radio button based on the wiring configuration used on port 2.
- **Xon/Xoff Flow Control:** Choose this selection if you have port 2 wired for Hardware Flow Control (Xon/Xoff) with RTS and CTS signal connected between all devices.
- **RTS Flow Control:** Choose this selection if you have Port 2 RTS signal wired between all devices.



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

## Network Slave Operation

This section describes how other devices on a network can communicate with a CPU port that you have configured as a *DirectNET* slave or MODBUS slave (DL06). A MODBUS host must use the MODBUS RTU protocol to communicate with the DL06 as a slave. The host software must send a MODBUS function code and MODBUS address to specify a PLC memory location the DL06 comprehends. The *DirectNET* host uses normal I/O addresses to access applicable DL06 CPU and system. No CPU ladder logic is required to support either MODBUS slave or *DirectNET* slave operation.

### MODBUS Function Codes Supported

The MODBUS function code determines whether the access is a read or a write, and whether to access a single data point or a group of them. The DL06 supports the MODBUS function codes described below.

MODBUS Function Code	Function	DL06 Data Types Available
<b>01</b>	Read a group of coils	Y, CR, T, CT
<b>02</b>	Read a group of inputs	X, SP
<b>05</b>	Set / Reset a single coil	Y, CR, T, CT
<b>15</b>	Set / Reset a group of coils Y,	CR, T, CT
<b>03, 04</b>	Read a value from one or more registers	V
<b>06</b>	Write a value into a single register	V
<b>16</b>	Write a value into a group of registers	V

### Determining the MODBUS Address

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

- By specifying the MODBUS data type and address
- By specifying a MODBUS address only

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

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

The actual equation used to calculate the address depends on the type of PLC data you are using. The PLC memory types are split into two categories for this purpose.

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

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

DL06 Memory Type	QTY (Dec.)	PLC Range(Octal)	MODBUS Address Range (Decimal)	MODBUS Data Type
<b>For Discrete Data Types .... Convert PLC Addr. to Dec. + Start of Range + Data Type</b>				
<b>Inputs (X)</b>	512	X0 – X777	2048 – 2559	Input
<b>Special Relays(SP)</b>	512	SP0 – SP777	3072 – 3583	Input
<b>Outputs (Y)</b>	512	Y0 – Y777	2048 – 2559	Coil
<b>Control Relays (CR)</b>	1024	C0 – C1777	3072 – 4095	Coil
<b>Timer Contacts (T)</b>	256	T0 – T377	6144 – 6399	Coil
<b>Counter Contacts (CT)</b>	128	CT0 – CT177	6400 – 6527	Coil
<b>Stage Status Bits(S)</b>	1024	S0 – S1777	5120 – 6143	Coil
<b>For Word Data Types .... Convert PLC Addr. to Dec. + Data Type</b>				
<b>Timer Current Values (V)</b>	256	V0 – V377	0 – 255	Input Register
<b>Counter Current Values (V)</b>	128	V1000 – V1177	512 – 639	Input Register
<b>V Memory, user data (V)</b>	3200	V1200 – V7377	640 – 3839	Holding Register
	4096	V10000 - V17777	4096 - 8191	Holding Register
<b>V Memory, non-volatile (V)</b>	128	V7400 – V7577	3840 – 3967	Holding Register

The following examples show how to generate the MODBUS address and data type for hosts which require this format.

### Example 1: V2100

Find the MODBUS address for User V location V2100.

Holding Reg 1088

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

V Memory, user data (V)	3200	V1200 – V7377	640 – 3839	Holding Register
-------------------------	------	---------------	------------	------------------

### Example 2: Y20

Find the MODBUS address for output Y20.

Coil 2064

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.

Outputs (V)	256	Y0 – Y377	2048 - 2303	Coil
-------------	-----	-----------	-------------	------

### Example 3: T10 Current Value

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

Input Reg. 8

1. Find Timer Current Values in the table.
2. Convert T10 into decimal (8).
3. Use the MODBUS data type from the table.

Timer Current Values (V)	128	V0 – V177	0 - 127	Input Register
--------------------------	-----	-----------	---------	----------------

### Example 4: C54

Find the MODBUS address for Control Relay C54.

Coil 3116

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.

Control Relays (CR)	512	C0 – C77	3072 – 3583	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, SP, Y, CR, S, T, C (contacts)
- Word – V, Timer current value, Counter current value

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

DL06 Memory Type	QTY (Dec.)	PLC Range (Octal)	MODBUS Address Range (Decimal)	484 Mode Address	584/984 Mode Address	MODBUS Data Type
<b>For Discrete Data Types ....Convert Addr. to Dec. + Start of Range + Data Type</b>						
<b>Inputs (X)</b>	512	X0 – X777	2048 – 2559	1001	100001	Input
<b>Special Relays (SP)</b>	512	SP0 – SP777	3072 – 3583	1001	100001	Input
<b>Outputs (Y)</b>	512	Y0 – Y777	2048 – 2559	1	1	Coil
<b>Control Relays (CR)</b>	1024	C0 – C1777	3072 – 4095	1	1	Coil
<b>Timer Contacts (T)</b>	256	T0 – T377	6144 – 6399	1	1	Coil
<b>Counter Contacts (CT)</b>	128	CT0 – CT177	6400 – 6527	1	1	Coil
<b>Stage Status Bits (S)</b>	1024	S0 – S1777	5120 – 6143	1	1	Coil
<b>For Word Data Types .... Convert Addr. to Dec. + Data Type</b>						
<b>Timer Current Values (V)</b>	256	V0 – V377	0 – 255	3001	30001	Input Register
<b>Counter Current Values (V)</b>	128	V1000 – V1177	512 – 639	3001	30001	Input Register
<b>V Memory, user data (V)</b>	3200	V1200 – V7377	640 – 3839	4001	40001	Holding Register
	4096	V10000 - V17777	4096 - 8191			Holding Register
<b>V-Memory, non-volatile (V)</b>	128	V7400 – V7577	3840 – 3967	4001	40001	Holding Register

The following examples show how to generate the MODBUS addresses for hosts which require this format.

### Example 1: V2100 584/984 Mode

Find the MODBUS address for User V location V2100.

41089

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

V Memory, system (V)	128	V1200 – V7377	3480 – 3735	4001	40001	Holding Register
----------------------	-----	---------------	-------------	------	-------	------------------

### Example 2: Y20 584/984 Mode

Find the MODBUS address for output Y20.

2065

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).

Outputs (V)	256	Y0 – Y377	2048 – 2303	1	1	Coil
-------------	-----	-----------	-------------	---	---	------

### Example 3: T10 Current Value 484 Mode

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

3009

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

Timer Current Values (V)	128	V0 – V177	0 – 127	3001	3001	Input Register
--------------------------	-----	-----------	---------	------	------	----------------

### Example 4: C54 584/984 Mode

Find the MODBUS address for Control Relay C54.

3117

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).

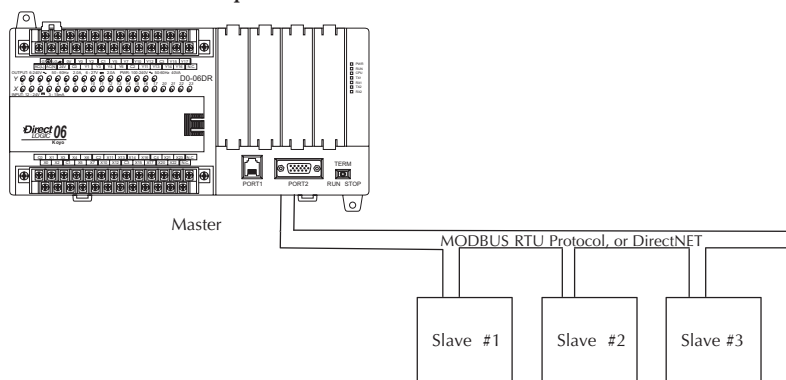
Control Relays(V)	512	C0 – C777	3072 – 3583	1	1	Coil
-------------------	-----	-----------	-------------	---	---	------

### Determining the DirectNET Address

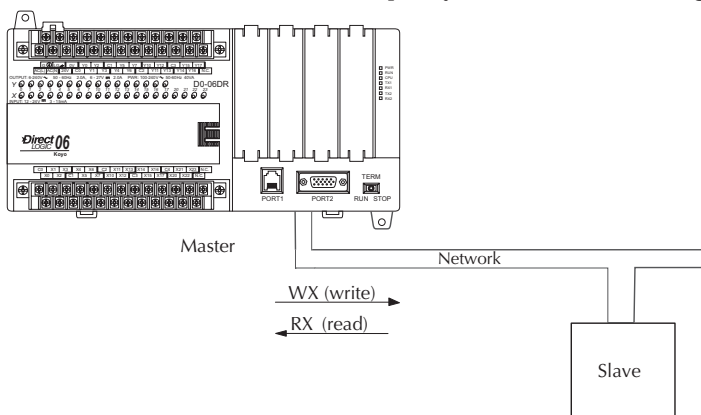
Addressing the memory types for DirectNET slaves is very easy. Use the ordinary native address of the slave device itself. To access a slave PLC's memory address V2000 via DirectNET, for example, the network master will request V2000 from the slave.

### Network Master Operation

This section describes how the DL06 can communicate on a MODBUS or *DirectNET* network as a master. For MODBUS networks, it uses the MODBUS RTU protocol, which must be interpreted by all the slaves on the network. Both MODBUS and *DirectNet* are single master/multiple slave networks. The master is the only member of the network that can initiate requests on the network. This section teaches you how to design the required ladder logic for network master operation.



When using the DL06 PLC as the master station, simple RLL instructions are used to initiate the requests. The WX instruction initiates network write operations, and the RX instruction initiates network read operations. Before executing either the WX or RX commands, we will need to load data related to the read or write operation onto the CPU's accumulator stack. When the WX or RX instruction executes, it uses the information on the stack combined with data in the instruction box to completely define the task, which goes to the port.

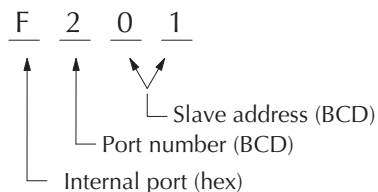


The following step-by-step procedure will provide you the information necessary to set up your ladder program to receive data from a network slave.



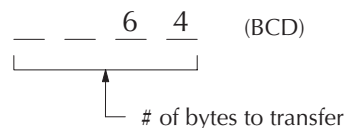
## Step 1: Identify Master Port # and Slave #

The first Load (LD) instruction identifies the communications port number on the network master (DL06) and the address of the slave station. This instruction can address up to 99 MODBUS slaves, or 90 *Direct*NET slaves. The format of the word is shown to the right. The “F2” in the upper byte indicates the use of the right port of the DL06 PLC, port number 2. The lower byte contains the slave address number in BCD (01 to 99).



## Step 2: Load Number of Bytes to Transfer

The second Load (LD) instruction determines the number of bytes which will be transferred between the master and slave in the subsequent WX or RX instruction. The value to be loaded is in BCD format (decimal), from 1 to 128 bytes.



The number of bytes specified also depends on the type of data you want to obtain. For example, the DL06 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 05 / 06 / 205 / 350 / 405 Memory	Bits per unit	Bytes
V memory	16	2
T / C current value	16	2
Inputs (X, SP)	8	1
Outputs (Y, C, Stage, T/C bits)	8	1
Scratch Pad Memory	8	1
Diagnostic Status	8	1
DL330 / 340 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	1
Diagnostic Status(5 word R/W)	16	10

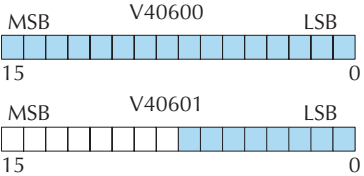
## Step 3: Specify Master Memory Area

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 DL06 CPU sends the number of bytes previously specified from its memory area beginning at the LDA address specified.

For an RX instruction, the DL06 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.

4 0 6 0 0 (octal)

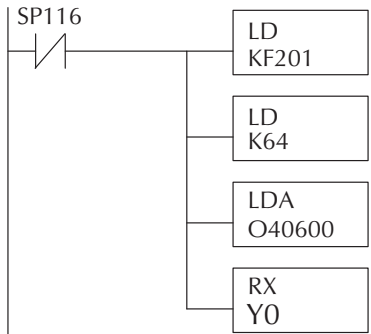


**NOTE:** Since V memory words are always 16 bits, you may not always use the whole word. For example, if you only specify 3 bytes and you are reading Y outputs from the slave, you will only get 24 bits of data. In this case, only the 8 least significant bits of the last word location will be modified.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

## Step 4: Specify Slave Memory Area

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

- **DirectNET** slaves – specify the same address in the WX and RX instruction as the slave's native I/O address
- **MODBUS DL405, DL205, or DL06** slaves – specify the same address in the WX and RX instruction as the slave's native I/O address
- **MODBUS 305** slaves – use the following table to convert DL305 addresses to MODBUS addresses



DL305 Series CPU Memory Type-to-MODBUS Cross Reference (excluding 350 CPU)

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

## Communications from a Ladder Program

Typically network communications will last longer than 1 scan. The program must wait for the communications to finish before starting the next transaction.

Port 2, which can be a master, has two Special Relay contacts associated with it (see Appendix D for comm port special relays). One indicates “Port busy” (SP116), and the other indicates “Port Communication Error” (SP117). The example above shows the use of these contacts for a network master that only reads a device (RX). The “Port Busy” bit is on while the PLC communicates with the slave. When the bit is off the program can initiate the next network request.

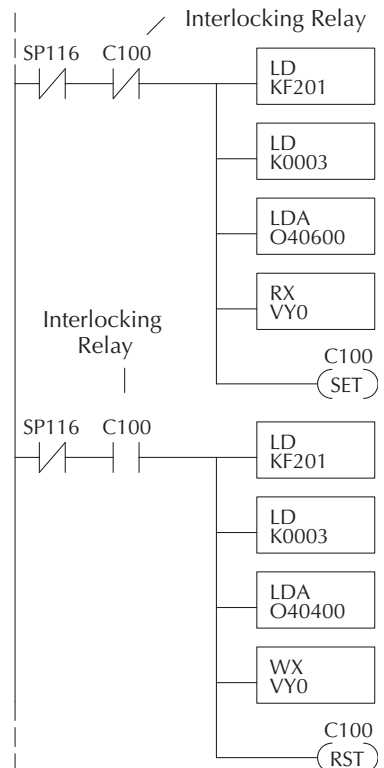
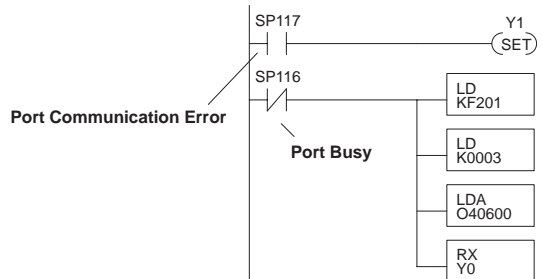
The “Port Communication Error” bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an RX or WX instruction is executed.

## Multiple Read and Write Interlocks

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

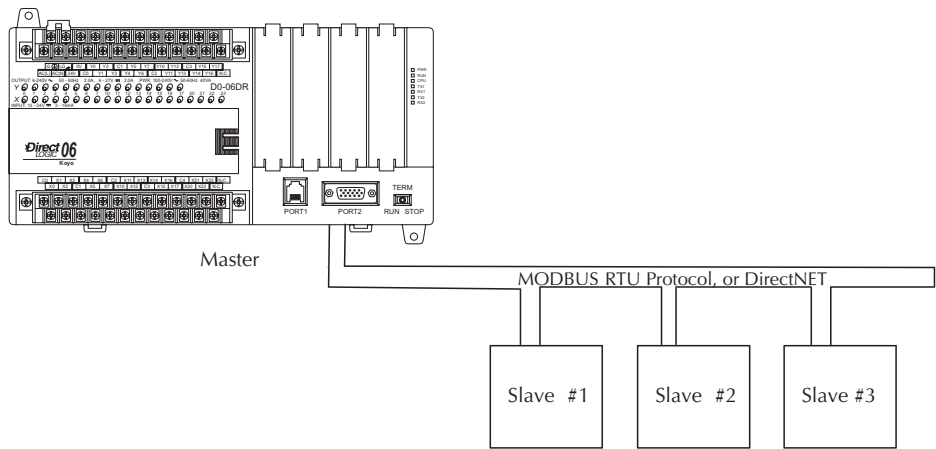
In the example to the right, after the RX instruction is executed, C0 is set. When the port has finished the communication task, the second routine is executed and C0 is reset.

If you're using RLL<sup>PLUS</sup> Stage Programming, you can put each routine in a separate program stage to ensure proper execution and switch from stage to stage allowing only one of them to be active at a time.



# Network Master Operation (using MRX and MWX Instructions)

This section describes how the DL06 can communicate on a MODBUS RTU network as a master using the MRX and MWX read/write instructions. These instructions allow you to enter native MODBUS addressing in your ladder logic program with no need to perform octal to decimal conversions. MODBUS is a single master/multiple slave network. The master is the only member of the network that can initiate requests on the network. This section teaches you how to design the required ladder logic for network master operation.



## MODBUS Function Codes Supported

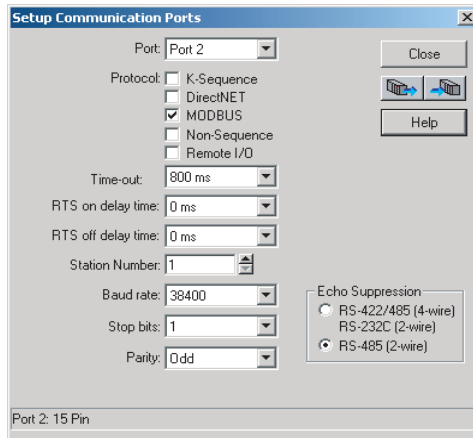
The MODBUS function code determines whether the access is a read or a write, and whether to access a single data point or a group of them. The DL06 supports the MODBUS function codes described below.

MODBUS Function Code	Function	DL06 Data Types Available
01	Read a group of coils	Y, CR, T, CT
02	Read a group of inputs	X, SP
05	Set / Reset a single coil (slave only)	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	Write a value into a single register (slave only)	V
07	Read Exception Status	V
08	Diagnostics	V
16	Write a value into a group of registers	V

## MODBUS Port Configuration

In *DirectSOFT32*, choose the PLC menu, then Setup, then “Secondary Comm Port”.

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



**Timeout:** Amount of time the port will wait after it sends a message to get a response before logging an error.

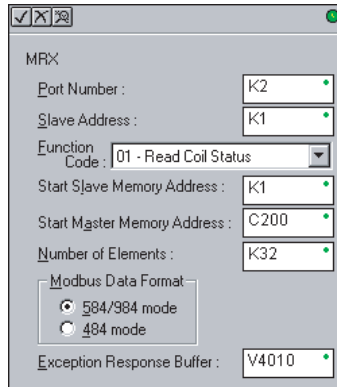
- **RTS On Delay Time:** The amount of time between raising the RTS line and sending the data.
- **RTS Off Delay Time:** The amount of time between resetting the RTS line after sending the data.
- **Station Number:** For making the CPU port a MODBUS master, choose “1”. The possible range for MODBUS slave numbers is from 1 to 247. Each slave must have a unique number. At powerup, the port is automatically a slave, unless and until the DL06 executes ladder logic MWX/MRX 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.
- **Echo Suppression:** Select the appropriate radio button based on the wiring configuration used on port 2.



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

### MODBUS Read from Network(MRX)

The MODBUS Read from Network (MRX) instruction is used by the DL06 network master to read a block of data from a connected slave device and to write the data into V-memory addresses within the master. The instruction allows the user to specify the MODBUS Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, MODBUS data format and the Exception Response Buffer.



- **Port Number:** must be DL06 Port 2 (K2)
- **Slave Address:** specify a slave station address (0–247)
- **Function Code:** The following MODBUS function codes are supported by the MRX instruction:
  - 01 – Read a group of coils
  - 02 – Read a group of inputs
  - 03 – Read holding registers
  - 04 – Read input registers
  - 07 – Read Exception status
  - 08 – Diagnostics
- **Start Slave Memory Address:** specifies the starting slave memory address of the data to be read. See the table on the following page.
- **Start Master Memory Address:** specifies the starting memory address in the master where the data will be placed. See the table on the following page.
- **Number of Elements:** specifies how many coils, input, holding registers or input register will be read. See the table on the following page.
- **MODBUS Data Format:** specifies MODBUS 584/984 or 484 data format to be used
- **Exception Response Buffer:** specifies the master memory address where the Exception Response will be placed. See the table on the following page.

## MRX Slave Memory Address

MRX Slave Address Ranges		
Function Code	MODBUS Data Format	Slave Address Range(s)
01 – Read Coil	484 Mode	1–999
01 – Read Coil	584/984 Mode	1–65535
02 – Read Input Status	484 Mode	1001–1999
02 – Read Input Status	584/984 Mode	10001–19999 (5 digit) or 100001–165535 (6 digit)
03 – Read Holding Register	484 Mode	4001–4999
03 – Read Holding Register	584/984	40001–49999 (5 digit) or 4000001–465535 (6 digit)
04 – Read Input Register	484 Mode	3001–3999
04 – Read Input Register	584/984 Mode	30001–39999 (5 digit) or 3000001–365535 (6 digit)
07 – Read Exception Status	484 and 584/984 Mode	n/a
08 – Diagnostics	484 and 584/984 Mode	0–65535

## MRX Master Memory Addresses

MRX Master Memory Address Ranges	
Operand Data Type	DL06 Range
Inputs X	0–1777
Outputs Y	0–1777
Control Relays C	0–3777
Stage Bits S	0–1777
Timer Bits T	0–377
Counter Bits CT	0–377
Special Relays SP	0–777
V–memory V	All
Global Inputs GX	0–3777
Global Outputs GY	0–3777

## MRX Number of Elements

Number of Elements	
Operand Data Type	DL06 Range
V–memory ..... V	All
Constant ..... K	1–2000

## MRX Exception Response Buffer

Exception Response Buffer	
Operand Data Type	DL06 Range
V–memory ..... V	All

### MODBUS Write to Network (MWX)

The MODBUS Write to Network (MWX) instruction is used to write a block of data from the network masters's (DL06) memory to MODBUS memory addresses within a slave device on the network. The instruction allows the user to specify the MODBUS Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, MODBUS data format and the Exception Response Buffer.

- **Port Number:** must be DL06 Port 2 (K2)
- **Slave Address:** specify a slave station address (0–247)
- **Function Code:** The following MODBUS function codes are supported by the MWX instruction:
  - 05 – Force Single coil
  - 06 – Preset Single Register
  - 08 – Diagnostics
  - 15 – Force Multiple Coils
  - 16 – Preset Multiple Registers
- **Start Slave Memory Address:** specifies the starting slave memory address where the data will be written.
- **Start Master Memory Address:** specifies the starting address of the data in the master that is to be written to the slave.
- **Number of Elements:** specifies how many consecutive coils or registers will be written to. This field is only active when either function code 15 or 16 is selected.
- **MODBUS Data Format:** specifies MODBUS 584/984 or 484 data format to be used.
- **Exception Response Buffer:** specifies the master memory address where the Exception Response will be placed.



## MWX Slave Memory Address

MWX Slave Address Ranges		
Function Code	MODBUS Data Format	Slave Address Range(s)
05 – Force Single Coil	484 Mode	1–999
05 – Force Single Coil	584/984 Mode	1–65535
06 – Preset Single Register	484 Mode	4001–4999
06 – Preset Single Register	84/984 Mode	40001–49999 (5 digit) or 400001–465535 (6 digit)
08 – Diagnostics	484 and 584/984 Mode	0–65535
15 – Force Multiple Coils	484	1–999
15 – Force Multiple Coils	585/984 Mode	1–65535
16 – Preset Multiple Registers	484 Mode	4001–4999
16 – Preset Multiple Registers	584/984 Mode	40001–49999 (5 digit) or 400001–465535 (6 digit)

## MWX Master Memory Addresses

MRX Master Memory Address Ranges		
Operand Data Type		DL06 Range
Inputs .....	X	0–777
Outputs .....	Y	0–777
Control Relays .....	C	0–1777
Stage Bits .....	S	0–1777
Timer Bits .....	T	0–377
Counter Bits .....	CT	0–177
Special Relays .....	SP	0–777
V-memory .....	V	All
Global Inputs .....	GX	0–3777
Global Outputs .....	GY	0–3777

## MWX Number of Elements

Number of Elements		
Operand Data Type		DL06 Range
V-memory .....	V	All
Constant .....	K	1–2000

## MWX Exception Response Buffer

Exception Response Buffer		
Operand Data Type		DL06 Range
V-memory .....	V	All

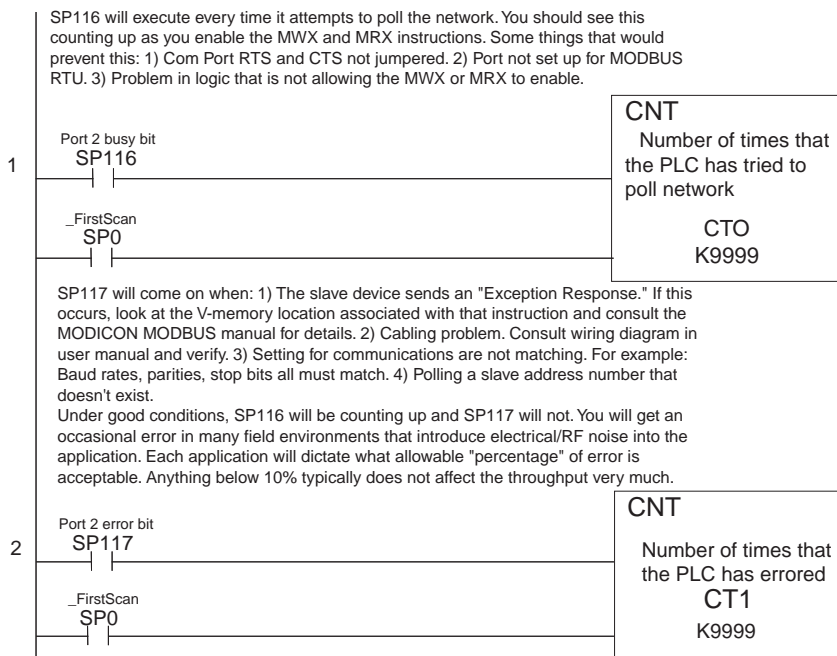
### MRX / MWX Example in DirectSOFT32

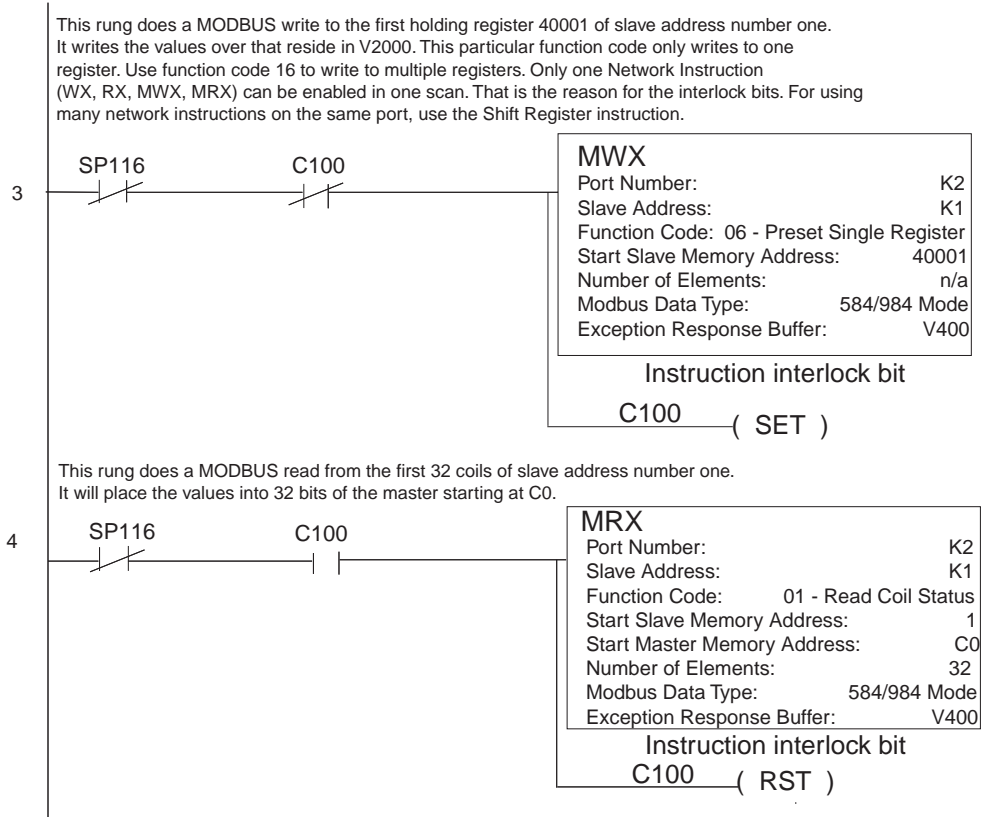
DL06 port 2 has two Special Relay contacts associated with it (see Appendix D for communication special relays). One indicates "Port busy" (SP116), and the other indicates "Port Communication Error" (SP117). The "Port Busy" bit is on while the PLC communicates with the slave. When the bit is off the program can initiate the next network request. The "Port Communication Error" bit turns on when the PLC has detected an error and use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an MRX or MWX instruction is executed. Typically network communications will last longer than 1 CPU scan. The program must wait for the communications to finish before starting the next transaction.

The "Port Communication Error" bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an RX or WX instruction is executed.

### Multiple Read and Write Interlocks

If you are using multiple reads and writes in the RLL program, you have to interlock the routines to make sure all the routines are executed. If you don't use the interlocks, then the CPU will only execute the first routine. This is because each port can only handle one transaction at a time. In the example below, after the RX instruction is executed, C100 is set. When the port has finished the communication task, the second routine is executed and C100 is reset. If you're using RLLPLUS Stage Programming, you can put each routine in a separate program stage to ensure proper execution and switch from stage to stage allowing only one of them to be active at a time.





# STANDARD RLL INSTRUCTIONS

---



# CHAPTER 5

## In This Chapter

Introduction . . . . .	5-2
Using Boolean Instructions . . . . .	5-5
Boolean Instructions . . . . .	5-10
Comparative Boolean . . . . .	5-26
Immediate Instructions . . . . .	5-32
Timer, Counter and Shift Register Instructions . . . . .	5-39
Accumulator / Stack Load and Output Data Instructions . . . . .	5-52
Logical Instructions (Accumulator) . . . . .	5-69
Math Instructions . . . . .	5-86
Transcendental Functions . . . . .	5-118
Bit Operation Instructions . . . . .	5-120
Number Conversion Instructions (Accumulator) . . . . .	5-127
Table Instructions . . . . .	5-141
Clock / Calendar Instructions . . . . .	5-171
CPU Control Instructions . . . . .	5-173
Program Control Instructions . . . . .	5-175
Interrupt Instructions . . . . .	5-183
Message Instructions . . . . .	5-186
MODBUS RTU Instructions . . . . .	5-201
ASCII Instructions . . . . .	5-207

# Introduction

DL06 Micro PLCs offer a wide variety of instructions to perform many different types of operations. This chapter shows you how to use each standard Relay Ladder Logic (RLL) instruction. In addition to these instructions, you may also need to refer to the Drum instruction in Chapter 6, or the Stage programming instructions in Chapter 7.

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

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

Instruction	Page	Instruction	Page
Accumulating Fast Timer (TMRAF)	5-42	And Store (AND STR)	5-16
Accumulating Timer (TMRA)	5-42	And with Stack (ANDS)	5-72
Add (ADD)	5-86	Arc Cosine Real (ACOSR)	5-119
Add Binary (ADDB)	5-99	Arc Sine Real (ASINR)	5-118
Add Binary Double (ADDBD)	5-100	Arc Tangent Real (ATANR)	5-119
Add Binary Top of Stack (ADDBS)	5-114	ASCII Clear Buffer (ACRB)	5-225
Add Double (ADDD)	5-87	ASCII Compare (CMPV)	5-217
Add Formatted (ADDF)	5-106	ASCII Constant (ACON)	5-187
Add Real (ADDR)	5-88	ASCII Extract (AEX)	5-216
Add to Top (ATT)	5-162	ASCII Find (AFIND)	5-213
Add Top of Stack (ADDS)	5-110	ASCII Input (AIN)	5-209
And (AND)	5-69	ASCII Print from V-memory (PRINTV)	5-223
And (AND)	5-31	ASCII Print to V-memory (VPRINT)	5-218
And (AND)	5-14	ASCII Swap Bytes (SWAPB)	5-224
AND Bit-of-Word (ANDB)	5-15	ASCII to HEX (ATH)	5-134
And Double (ANDD)	5-70	Binary (BIN)	5-127
And Formatted (ANDF)	5-71	Binary Coded Decimal (BCD)	5-128
And If Equal (ANDE)	5-28	Binary to Real Conversion (BTOR)	5-131
And If Not Equal (ANDNE)	5-28	Compare (CMP)	5-81
And Immediate (ANDI)	5-33	Compare Double (CMPD)	5-82
AND Move (ANDMOV)	5-167	Compare Formatted (CMPF)	5-83
And Negative Differential (ANDND)	5-22	Compare Real Number (CMPR)	5-85
And Not (ANDN)	5-31	Compare with Stack (CMPS)	5-84
And Not (ANDN)	5-14	Cosine Real (COSR)	5-118
And Not Bit-of-Word (ANDNB)	5-15	Counter (CNT)	5-45
And Not Immediate (ANDNI)	5-33	Data Label (DLBL)	5-187
And Positive Differential (ANDPD)	5-22	Date (DATE)	5-171

Instruction	Page	Instruction	Page
Decode (DECO)	5-126	Load Accumulator Indexed (LDX)	5-61
Decrement (DEC)	5-98	Load Accumulator Indexed from Data Constants (LDSX)	5-62
Decrement Binary (DECB)	5-105	Load Address (LDA)	5-60
Degree Real Conversion (DEGR)	5-133	Load Double (LDD)	5-58
Disable Interrupts (DISI)	5-184	Load Formatted (LDF)	5-59
Divide (DIV)	5-95	Load Immediate (LDI)	5-37
Divide Binary (DIVB)	5-104	Load Immediate Formatted (LDIF)	5-38
Divide Binary by Top OF Stack (DIVBS)	5-117	Load Label (LDLBL)	5-142
Divide by Top of Stack (DIVS)	5-113	Load Real Number (LDR)	5-63
Divide Double (DIVD)	5-96	Master Line Reset (MLR)	5-181
Divide Formatted (DIVF)	5-109	Master Line Set (MLS)	5-181
Divide Real (DIVR)	5-97	MODBUS Read from Network (MRX)	5-201
Enable Interrupts (ENI)	5-183	MODBUS Write to Network (MWX)	5-204
Encode (ENCO)	5-125	Move (MOV)	5-141
End (END)	5-173	Move Memory Cartridge (MOVMC)	5-142
Exclusive Or (XOR)	5-77	Multiply (MUL)	5-92
Exclusive Or Double (XORD)	5-78	Multiply Binary (MULB)	5-103
Exclusive Or Formatted (XORF)	5-79	Multiply Binary Top of Stack (MULBS)	5-116
Exclusive OR Move (XORMOV)	5-167	Multiply Double (MULD)	5-93
Exclusive Or with Stack (XORS)	5-80	Multiply Formatted (MULF)	5-108
External Interrupt Program Example	5-184	Multiply Real (MULR)	5-94
Fault (FAULT)	5-186	Multiply Top of Stack (MULS)	5-112
Fill (FILL)	5-146	No Operation (NOP)	5-173
Find (FIND)	5-147	Not (NOT)	5-19
Find Block (FINDB)	5-169	Numerical Constant (NCON)	5-187
Find Greater Than (FDGT)	5-148	Or (OR)	5-73
For / Next (FOR) (NEXT)	5-176	Or (OR)	5-30
Goto Label (GOTO) (LBL)	5-175	Or (OR)	5-12
Goto Subroutine (GTS) (SBR)	5-178	Or Bit-of-Word (ORB)	5-13
Gray Code (GRAY)	5-138	Or Double (ORD)	5-74
HEX to ASCII (HTA)	5-135	Or Formatted (ORF)	5-75
Increment (INC)	5-98	Or If Equal (ORE)	5-27
Increment Binary (INCB)	5-105	Or Immediate (ORI)	5-32
Interrupt (INT)	5-183	OR Move (ORMOV)	5-167
Interrupt Return (IRT)	5-183	Or Negative Differential (ORND)	5-21
Interrupt Return Conditional (IRTC)	5-183	Or Not (ORN)	5-30
Invert (INV)	5-129	Or Not (ORN)	5-12
LCD	5-197	Or Not Bit-of-Word (ORNB)	5-13
Load (LD)	5-57	Or Not Immediate (ORNI)	5-32

## Chapter 5: Standard RLL Instructions

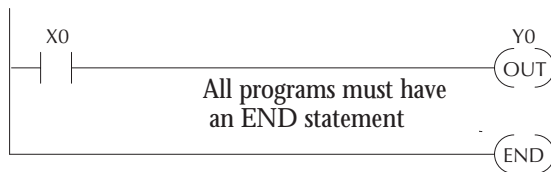
Instruction	Page	Instruction	Page
OR Not Immediate Instructions Cont'd	5-33	Shuffle Digits (SFLDGT)	5-139
Or Out (OR OUT)	5-17	Sine Real (SINR)	5-118
Or Out Immediate (OROUTI)	5-34	Source to Table (STT)	5-156
Or Positive Differential (ORPD)	5-21	Square Root Real (SQRTR)	5-119
Or Store (OR STR)	5-16	Stage Counter (SGCNT)	5-47
Or with Stack (ORS)	5-76	Stop (STOP)	5-173
Out (OUT)	5-64	Store (STR)	5-29
Out (OUT)	5-17	Store (STR)	5-10
Out Bit-of-Word (OUTB)	5-18	Store Bit-of-Word (STRB)	5-11
Out Double (OUTD)	5-64	Store If Equal (STRE)	5-26
Out Formatted (OUTF)	5-65	Store If Not Equal (STRNE)	5-26
Out Immediate (OUTI)	5-34	Store Immediate (STRI)	5-32
Out Immediate Formatted (OUTIF)	5-35	Store Negative Differential (STRND)	5-20
Out Indexed (OUTX)	5-67	Store Not (STRN)	5-29
Out Least (OUTL)	5-68	Store Not (STRN)	5-10
Out Most (OUTM)	5-68	Store Not Bit-of-Word (STRNB)	5-11
Pause (PAUSE)	5-25	Store Not Immediate (STRNI)	5-32
Pop (POP)	5-65	Store Positive Differential (STRPD)	5-20
Positive Differential (PD)	5-19	Subroutine Return (RT)	5-178
Print Message (PRINT)	5-189	Subroutine Return Conditional (RTC)	5-178
Radian Real Conversion (RADR)	5-133	Subtract (SUB)	5-89
Read from Network (RX)	5-193	Subtract Binary (SUBB)	5-101
Real to Binary Conversion (RTOB)	5-132	Subtract Binary Double (SUBBD)	5-102
Remove from Bottom (RFB)	5-153	Subtract Binary Top of Stack (SUBBS)	5-115
Remove from Table (RFT)	5-159	Subtract Double (SUBD)	5-90
Reset (RST)	5-23	Subtract Formatted (SUBF)	5-107
Reset Bit-of-Word (RSTB)	5-24	Subtract Real (SUBR)	5-91
Reset Immediate (RSTI)	5-36	Subtract Top of Stack (SUBS)	5-111
Reset Watch Dog Timer (RSTWT)	5-174	Sum (SUM)	5-120
Rotate Left (ROTL)	5-123	Swap (SWAP)	5-170
Rotate Right (ROTR)	5-124	Table Shift Left (TSHFL)	5-165
RSTBIT	5-144	Table Shift Right (TSHFR)	5-165
Segment (SEG)	5-137	Table to Destination (TTD)	5-150
Set (SET)	5-23	Tangent Real (TANR)	5-118
Set Bit-of-Word (SETB)	5-24	Ten's Complement (BCDCPL)	5-130
Set Immediate (SETI)	5-36	Time (TIME)	5-172
SETBIT	5-144	Timer (TMR) and Timer Fast (TMRF)	5-40
Shift Left (SHFL)	5-121	Understanding Master Control Relays	5-181
Shift Register (SR)	5-51	Up Down Counter (UDC)	5-49
Shift Right (SHFR)	5-122	Write to Network (WX)	5-195

## Using Boolean Instructions

Do you ever wonder why so many PLC manufacturers always quote the scan time for a 1K boolean program? Simple. 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. Since the *DirectSOFT32* package allows you to use graphic symbols to build the program, you don't absolutely have to know the mnemonics of the instructions. However, it may be helpful at some point, especially if you ever have to troubleshoot the program with a Handheld Programmer. The following paragraphs show how these instructions are used to build simple ladder programs.

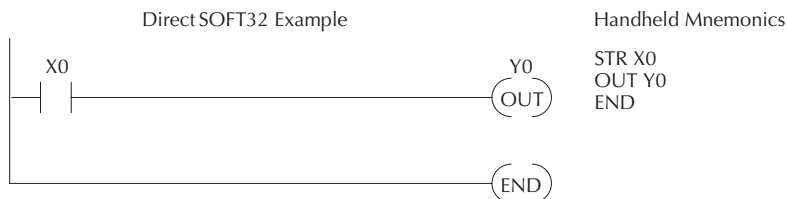
### END Statement

All DL06 programs require an END statement as the last instruction. This tells the CPU that this is the end of the program. Normally, any instructions placed after the END statement will not be executed. There are exceptions to this such as interrupt routines, etc. Chapter 5 discusses the instruction set in detail.



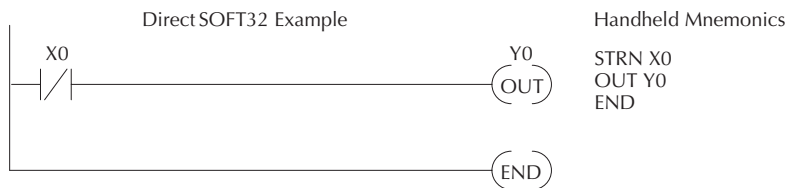
### Simple Rungs

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



### Normally Closed Contact

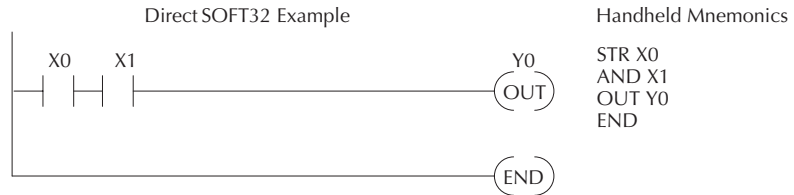
Normally closed contacts are also very common. This is accomplished with the Store Not, or STRN instruction. The following example shows a simple rung with a normally closed contact.





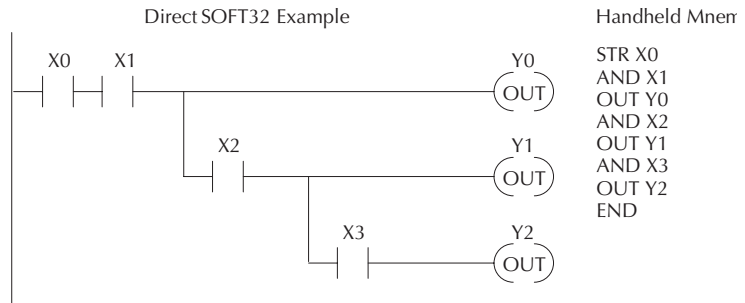
### Contacts in Series

Use the AND instruction to join two or more contacts in series. The following example shows two contacts in series and a single output coil. The instructions used would be STR X0, AND X1, followed by OUT Y0.

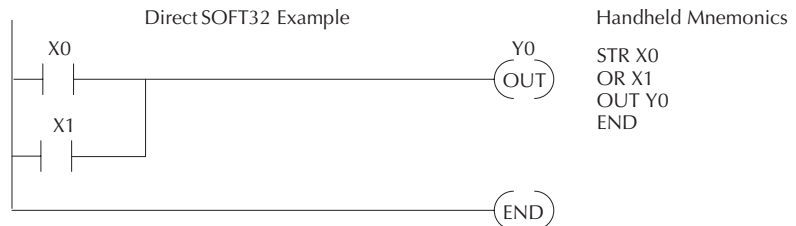


### Midline Outputs

Sometimes it is necessary to use midline outputs to get additional outputs that are conditional on other contacts. The following example shows how you can use the AND instruction to continue a rung with more conditional outputs.



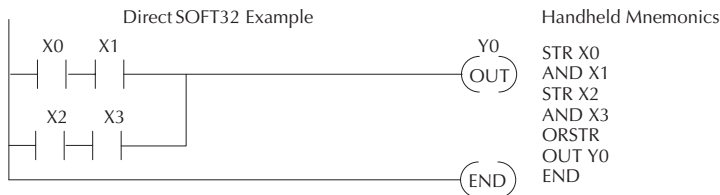
### Parallel Elements



You may also have to join contacts in parallel. The OR instruction allows you to do this. The following example shows two contacts in parallel and a single output coil. The instructions would be STR X0, OR X1, followed by OUT Y0.

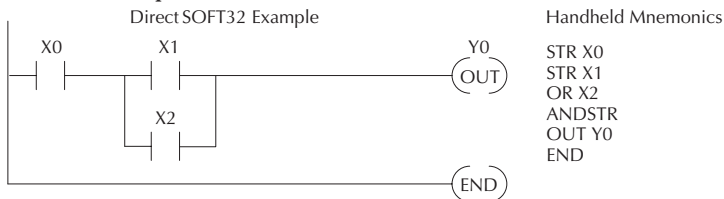
## Joining Series Branches in Parallel

Quite often it is necessary to join several groups of series elements in parallel. The Or Store (ORSTR) instruction allows this operation. The following example shows a simple network consisting of series elements joined in parallel.



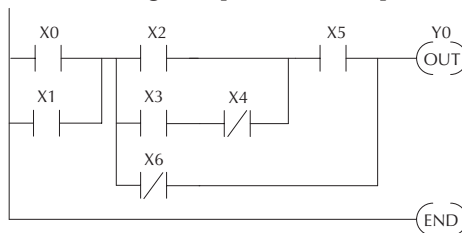
## Joining Parallel Branches in Series

You can also join one or more parallel branches in series. The And Store (ANDSTR) instruction allows this operation. The following example shows a simple network with contact branches in series with parallel contacts.



## Combination Networks

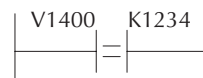
You can combine the various types of series and parallel branches to solve most any application problem. The following example shows a simple combination network.



## Comparative Boolean

Some PLC manufacturers make it really difficult to do a simple comparison of two numbers. Some of them require you to move the data all over the place before you can actually perform the comparison. The DL06 Micro PLCs provide Comparative Boolean instructions that allow you to quickly and easily solve this problem. The Comparative Boolean provides evaluation of two 4-digit values using boolean contacts. The valid evaluations are: equal to, not equal to, equal to or greater than, and less than.

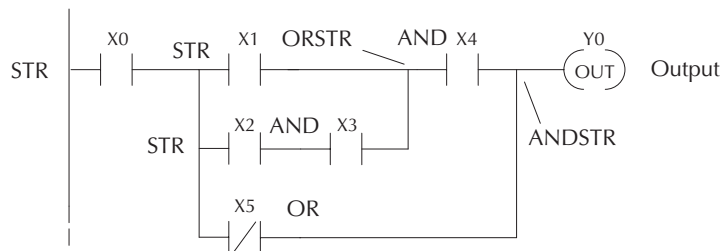
In the following example when the value in V-memory location V1400 is equal to the constant value 1234, Y3 will energize.



Boolean Stack

There are limits to how many elements you can include in a rung. This is because the DL06 PLCs use an 8-level boolean stack to evaluate the various logic elements. The boolean stack is a temporary storage area that solves the logic for the rung. Each time the program encounters a STR instruction, the instruction is placed on the top of the stack. Any other STR instructions already on the boolean stack are pushed down a level. The ANDSTR, and ORSTR instructions combine levels of the boolean stack when they are encountered. An error will occur during program compilation if the CPU encounters a rung that uses more than the eight levels of the boolean stack.

The following example shows how the boolean stack is used to solve boolean logic.



STR X0

1	STR X0
2	
3	
4	
5	
6	
7	
8	

STR X1

1	STR X1
2	STR X0
3	
4	
5	
6	
7	
8	

STR X2

1	STR X2
2	STR X1
3	STR X0
4	
5	
6	
7	
8	

AND X3

1	X2 AND X3
2	STR X1
3	STR X0
4	
5	
6	
7	
8	

ORSTR

1	X1 or (X2 AND X3)
2	STR X0
3	

AND X4

1	X4 AND {X1 or (X2 AND X3)}
2	STR X0
3	

ORNOT X5

1	NOT X5 OR X4 AND {X1 OR (X2 AND X3)}
2	STR X0
3	

8	
---	--

8	
---	--

8	
---	--

ANDSTR

1	X0 AND (NOT X5 or X4) AND {X1 or (X2 AND X3)}
2	
3	

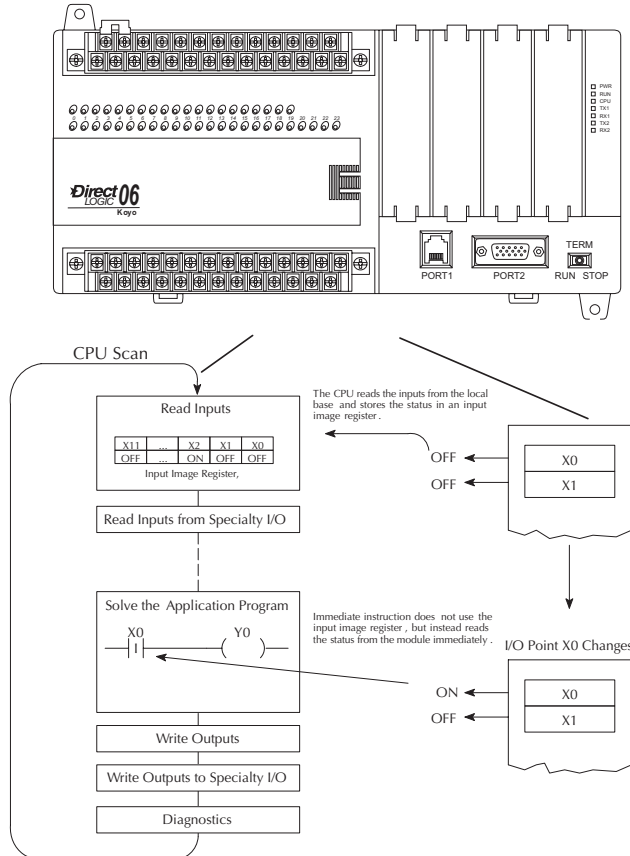
8	
---	--

## Immediate Boolean

The DL06 Micro PLCs can usually complete an operation cycle in a matter of milliseconds. However, in some applications you may not be able to wait a few milliseconds until the next I/O update occurs. The DL06 PLCs offer Immediate input and outputs which are special boolean instructions that allow reading directly from inputs and writing directly to outputs during the program execution portion of the CPU cycle. You may recall that this is normally done during the input or output update portion of the CPU cycle. The immediate instructions take longer to execute because the program execution is interrupted while the CPU reads or writes the I/O point. This function is not normally done until the read inputs or the write outputs portion of the CPU cycle.



**NOTE:** Even though the immediate input instruction reads the most current status from the input point, it only uses the results to solve that one instruction. It does not use the new status to update the image register. Therefore, any regular instructions that follow will still use the image register values. Any immediate instructions that follow will access the I/O again to update the status. The immediate output instruction will write the status to the I/O and update the image register.



# Boolean Instructions

## Store (STR)

The Store instruction begins a new rung or an additional branch in a rung with a normally open contact. Status of the contact will be the same state as the associated image register point or memory location.



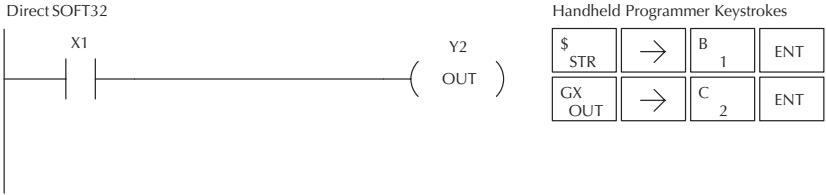
## Store Not (STRN)

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

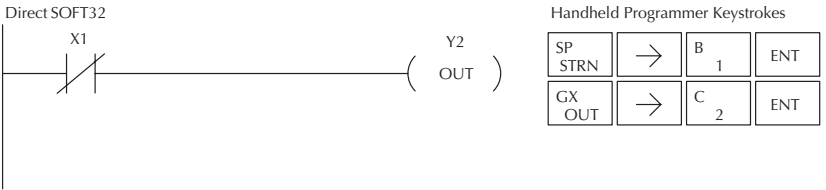


Operand Data Type	DL06 Range
..... A	aaa
Inputs..... X	0-777
Outputs..... Y	0-777
Control Relays..... C	0-1777
Stage..... S	0-1777
Timer..... T	0-377
Counter C..... CT	0-177
Special Relay..... SP	0-777

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

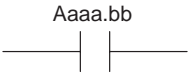


In the following Store Not example, when input X1 is off output Y2 will energize.



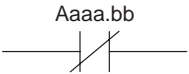
Store Bit-of-Word (STRB)

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



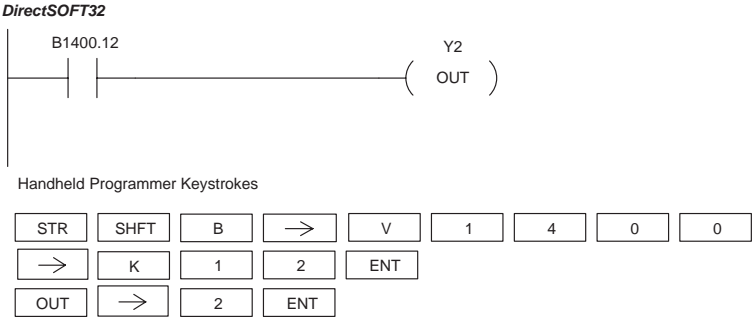
Store Not Bit-of-Word (STRNB)

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

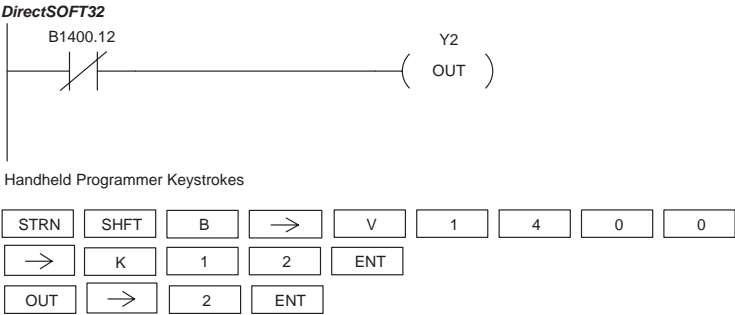


Operand Data Type		DL06 Range	
	A	aaa	bb
V memory	B	See memory map	BCD, 0 to 15
Pointer	PB	See memory map	BCD, 0 to 15

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

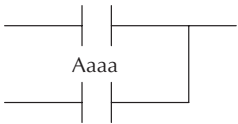


In the following Store Not Bit-of-Word example, when bit 12 of V-memory location V1400 is off, output Y2 will energize.



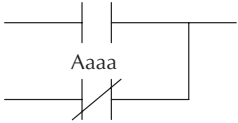
Or (OR)

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



Or Not (ORN)

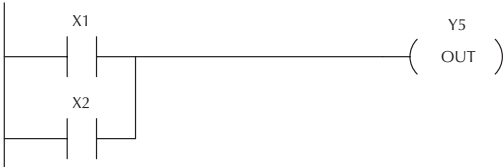
The Or Not instruction logically ors a normally closed contact in parallel with another contact in a rung. The status of the contact will be opposite the state of the associated image register point or memory location.



Operand Data Type	DL06 Range
..... A	<b>aaa</b>
Inputs ..... X	0-777
Outputs ..... Y	0-777
Control Relays ..... C	0-1777
Stage ..... S	0-1777
Timer ..... T	0-377
Counter ..... CT	0-177
Special Relay ..... SP	0-777

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

Direct SOFT32



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
Q OR	→	C 2	ENT
GX OUT	→	F 5	ENT

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

Direct SOFT32

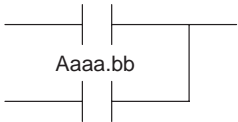


Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
R ORN	→	C 2	ENT
GX OUT	→	F 5	ENT

### Or Bit-of-Word (ORB)

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.



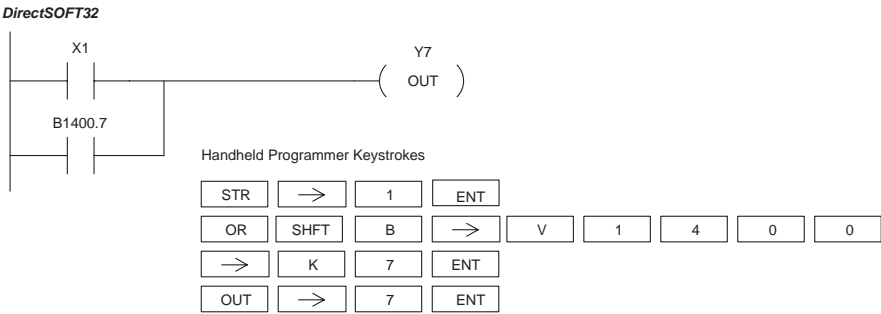
### Or Not Bit-of-Word (ORNB)

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.

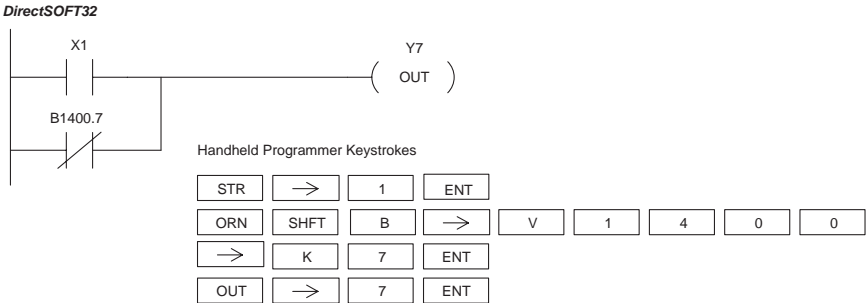


Operand Data Type		DL06 Range	
.....	A	<b>aaa</b>	<b>bb</b>
V memory .....	B	See memory map	BCD, 0 to 15
Pointer .....	PB	See memory map	BCD, 0 to 15

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



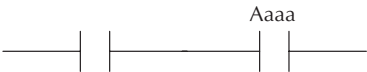
In the following Or Bit-of-Word example, when input X1 is on or bit 7 of V1400 is off, output Y5 will energize.





And (AND)

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



And Not (ANDN)

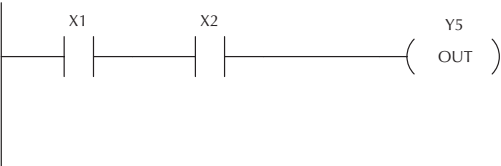
The And Not instruction logically ands a normally closed contact in series with another contact in a rung. The status of the contact will be opposite the state of the associated image register point or memory location.



Operand Data Type	DL06 Range
..... <b>A</b>	<b>aaa</b>
Inputs ..... X	0-777
Outputs ..... Y	0-777
Control Relays ..... C	0-1777
Stage ..... S	0-1777
Timer ..... T	0-377
Counter ..... CT	0-177
Special Relay ..... SP	0-777

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

DirectSOFT32

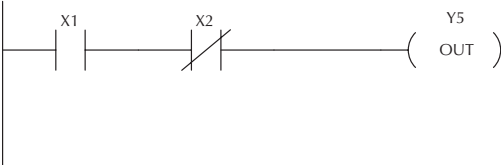


Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
V AND	→	C 2	ENT
GX OUT	→	F 5	ENT

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

DirectSOFT32



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
W ANDN	→	C 2	ENT
GX OUT	→	F 5	ENT

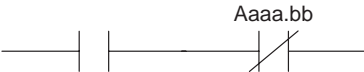
### AND Bit-of-Word (ANDB)

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



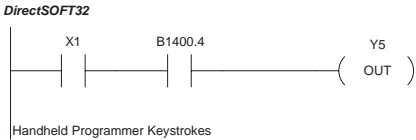
### And Not Bit-of-Word (ANDNB)

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



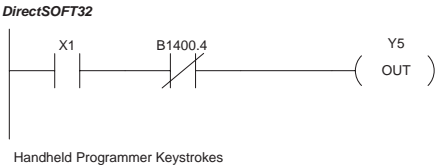
Operand Data Type		DL06 Range	
.....	A	<b>aaa</b>	<b>bb</b>
V memory.....	B	See memory map	BCD, 0 to 15
Pointer .....	PB	See memory map	BCD, 0 to 15

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



STR	→	1	ENT					
AND	SHFT	B	→	V	1	4	0	0
→	K	4	ENT					
OUT	→	5	ENT					

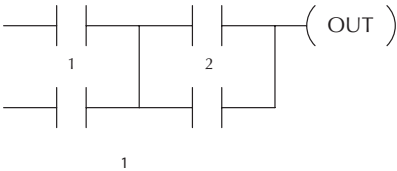
In the following And Not Bit-of-Word example, when input X1 is on and bit 4 of V1400 is off output Y5 will energize.



STR	→	1	ENT					
ANDN	SHFT	B	→	V	1	4	0	0
→	K	4	ENT					
OUT	→	5	ENT					

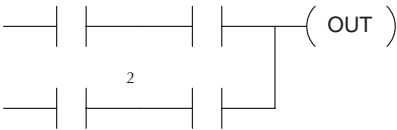
And Store (AND STR)

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

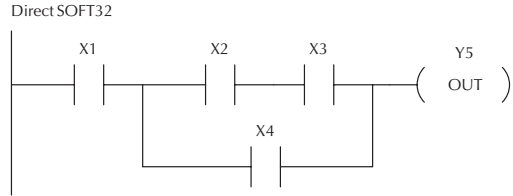


Or Store (OR STR)

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



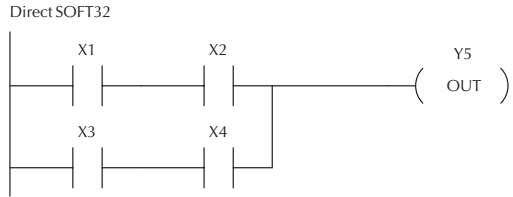
In the following And Store example, the branch consisting of contacts X2, X3, and X4 have been anded with the branch consisting of contact X1.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
\$ STR	→	C 2	ENT
V AND	→	D 3	ENT
Q OR	→	E 4	ENT
L ANDST	ENT		
GX OUT	→	F 5	ENT

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

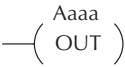


Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
V AND	→	C 2	ENT
\$ STR	→	D 3	ENT
V AND	→	E 4	ENT
M ORST	ENT		
GX OUT	→	F 5	ENT

# Out (OUT)

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

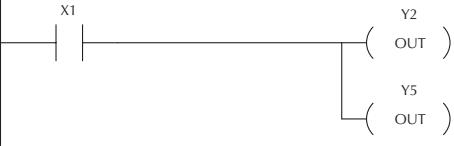


Multiple Out instructions referencing the same discrete location should not be used since only the last Out instruction in the program will control the physical output point. Instead, use the next instruction, the Or Out.

Operand Data Type	DL06 Range
..... <b>A</b>	<b>aaa</b>
Inputs ..... X	0-777
Outputs ..... Y	0-777
Control Relays ..... C	0-1777

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

DirectSOFT32

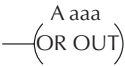


Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
GX OUT	→	C 2	ENT
GX OUT	→	F 5	ENT

# Or Out (OR OUT)

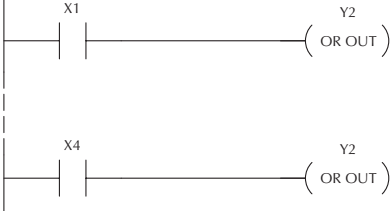
The Or Out instruction allows more than one rung of discrete logic to control a single output. Multiple Or Out instructions referencing the same output coil may be used, since *all* contacts controlling the output are logically ORED together. If the status of *any* rung is on, the output will also be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



Operand Data Type	DL06 Range
..... <b>A</b>	<b>aaa</b>
Inputs ..... X	0-777
Outputs ..... Y	0-777
Control Relays ..... C	0-1777

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

DirectSOFT32

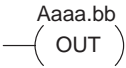


Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
O INST#	D 3	F 5	ENT
\$ STR	→	E 4	ENT
O INST#	D 3	F 5	ENT

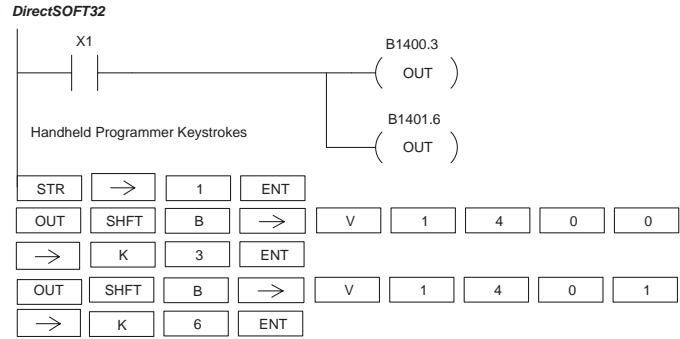
Out Bit-of-Word (OUTB)

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.

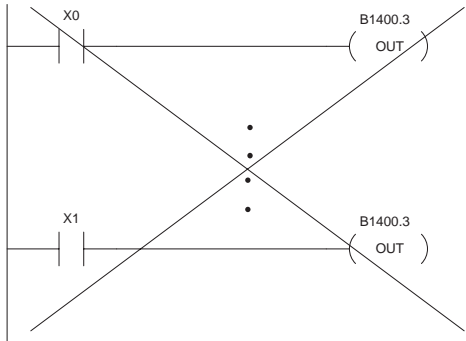


Operand Data Type		DL06 Range	
	A	aaa	bb
V memory	B	See memory map	BCD, 0 to 15
Pointer	PB	See memory map	BCD, 0 to 15

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

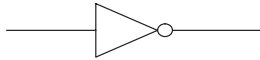


The following Out Bit-of-Word example contains two Out Bit-of-Word instructions using the same bit in the same memory word. The final state bit 3 of V1400 is ultimately controlled by the last rung of logic referencing it. X1 will override the logic state controlled by X0. To avoid this situation, multiple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



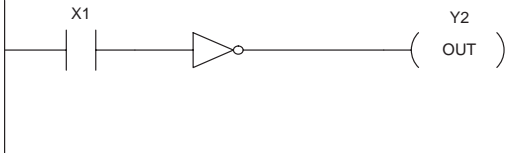
## Not (NOT)

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



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

DirectSOFT32



Handheld Programmer Keystrokes

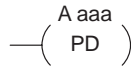
\$ STR	→	B 1	ENT
SHFT	N TMR	O INST#	T MLR
GX OUT	→	C 2	ENT



**NOTE:** DirectSOFT 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 DirectSOFT versions earlier than 1.1i.

## Positive Differential (PD)

The Positive Differential instruction is typically known as a one shot. When the input logic produces an off to on transition, the output will energize for one CPU scan.



Operand Data Type	DL06 Range
..... A	aaa
Inputs ..... X	0-777
Outputs ..... Y	0-777
Control Relays ..... C	0-1777

In the following example, every time X1 makes an off to on transition, C0 will energize for one scan.

DirectSOFT32



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
SHFT	P CV	SHFT	D 3
		→	A 0

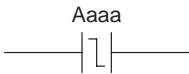
Store Positive Differential (STRPD)

The Store Positive Differential instruction begins a new rung or an additional branch in a rung with a contact. The contact closes for one CPU scan when the state of the associated image register point makes an Off-to-On transition. Thereafter, the contact remains open until the next Off-to-On transition (the symbol inside the contact represents the transition). This function is sometimes called a “one-shot”.



Store Negative Differential (STRND)

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



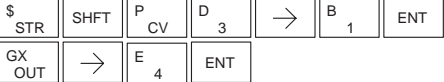
Operand Data Type	DL06 Range
..... <b>A</b>	<b>aaa</b>
Inputs .....	X
Outputs .....	Y
Control Relays .....	C
Stage .....	S
Timer .....	T
Counter .....	CT
	0-777
	0-777
	0-1777
	0-1777
	0-377
	0-177

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

DirectSOFT32



Handheld Programmer Keystrokes

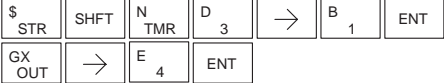


In the following example, each time X1 is makes an On-to-Off transition, Y4 will energize for one scan.

DirectSOFT32

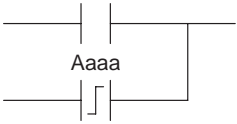


Handheld Programmer Keystrokes



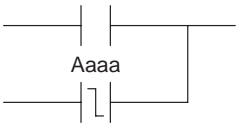
Or Positive Differential (ORPD)

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



Or Negative Differential (ORND)

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



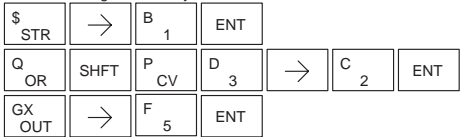
Operand Data Type	DL06 Range
..... A	aaa
Inputs ..... X	0-777
Outputs ..... Y	0-777
Control Relays ..... C	0-1777
Stage ..... S	0-1777
Timer ..... T	0-377
Counter ..... CT	0-177

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

DirectSOFT32



Handheld Programmer Keystrokes

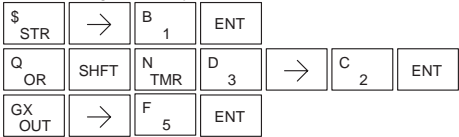


In the following example, Y 5 will energize whenever X1 is on, or for one CPU scan when X2 transitions from On to Off.

DirectSOFT32



Handheld Programmer Keystrokes





And Positive Differential (ANDPD)

The And Positive Differential instruction logically ands 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.



And Negative Differential (ANDND)

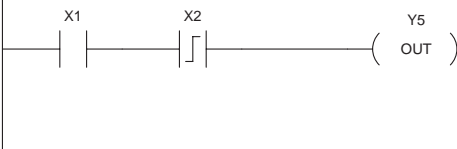
The And Negative Differential instruction logically ands a normally open contact in parallel with another contact in a rung. The status of the contact will be open until the associated image register point makes an On-to-Off transition, closing it for one CPU scan. Thereafter, it remains open until another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



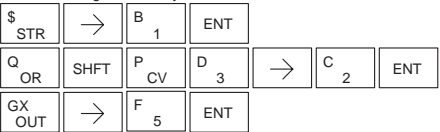
Operand Data Type	DL06 Range
..... A	aaa
Inputs ..... X	0-777
Outputs ..... Y	0-777
Control Relays ..... C	0-1777
Stage ..... S	0-1777
Timer ..... T	0-377
Counter ..... CT	0-177

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

DirectSOFT32

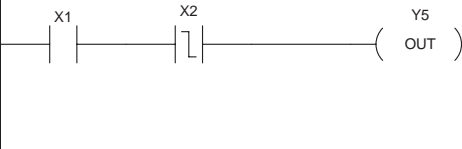


Handheld Programmer Keystrokes

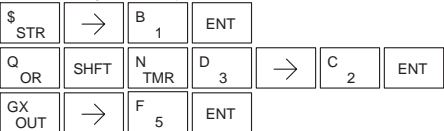


In the following example, Y5 will energize for one CPU scan whenever X1 is on and X2 transitions from On to Off.

DirectSOFT32

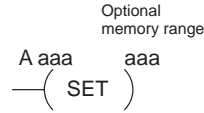


Handheld Programmer Keystrokes



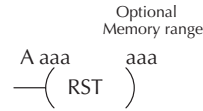
## Set (SET)

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



## Reset (RST)

The Reset instruction 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.



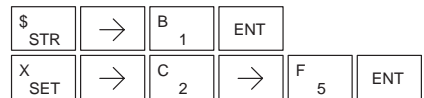
Operand Data Type	DL06 Range
..... A	aaa
Inputs ..... X	0-777
Outputs ..... Y	0-777
Control Relays ..... C	0-1777
Stage ..... S	0-1777
Timer ..... T	0-377
Counter ..... CT	0-177

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

DirectSOFT32



Handheld Programmer Keystrokes

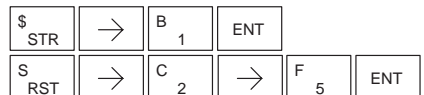


In the following example when X1 is on, Y2 through Y5 will be reset or de-energized.

DirectSOFT32

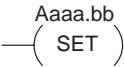


Handheld Programmer Keystrokes



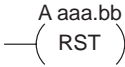
Set Bit-of-Word (SETB)

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



Reset Bit-of-Word (RSTB)

The Reset Bit-of-Word instruction resets or turns off a bit in a V memory location. Once the bit is reset it is not necessary for the input controlling the Reset Bit-of-Word instruction to remain on.



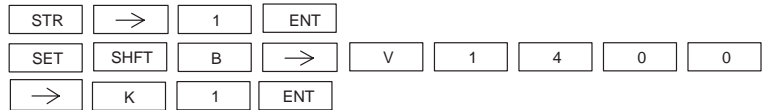
Operand Data Type		DL06 Range	
..... A		aaa	bb
V memory .....	B	See memory map	BCD, 0 to 15
Pointer .....	PB	See memory map	BCD, 0 to 15

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

DirectSOFT32



Handheld Programmer Keystrokes

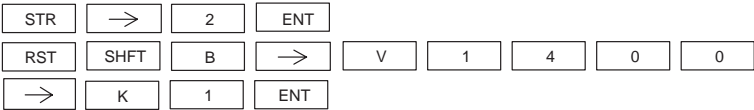


In the following example when X2 turns on, bit 1 in V1400 is reset to the off state.

DirectSOFT32

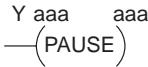


Handheld Programmer Keystrokes



# Pause (PAUSE)

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



Operand Data Type	DL06 Range
	<b>aaa</b>
Outputs ..... Y	0-777

In the following example, when X1 is ON, Y5–Y7 will be turned OFF. The execution of the ladder program will not be affected.

DirectSOFT32



Since the D2–HPP Handheld Programmer does not have a specific Pause key, you can use the corresponding instruction number for entry (#960), or type each letter of the command.

Handheld Programmer Keystrokes

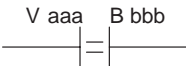


In some cases, you may want certain output points in the specified pause range to operate normally. In that case, use Aux 58 to over-ride the Pause instruction.

# Comparative Boolean

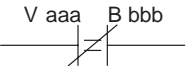
## Store If Equal (STRE)

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



## Store If Not Equal (STRNE)

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



Operand Data Type		DL06 Range	
		aaa	bbb
V memory	B	See memory map	See memory map
Pointer	V	See memory map	See memory map
Constant	P	See memory map	See memory map
	K	—	0-9999

In the following example, when the value in V memory location V2000 = 4933 , Y3 will energize.

DirectSOFT32



Handheld Programmer Keystrokes

\$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0
→	E 4	J 9	D 3	D 3	ENT		
GX OUT	→	D 3	ENT				

In the following example, when the value in V memory location V2000 ≠ 5060, Y3 will energize.

DirectSOFT32

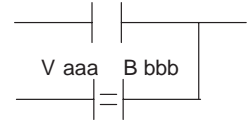


Handheld Programmer Keystrokes

SP STRN	SHFT	E 4	→	C 2	A 0	A 0	A 0
→	F 5	A 0	G 6	A 0	ENT		
GX OUT	→	D 3	ENT				

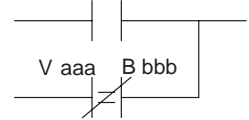
## Or If Equal (ORE)

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



## Or If Not Equal (ORNE)

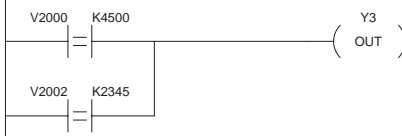
The Or If Not Equal instruction connects a normally closed comparative contact in parallel with another contact. The contact will be on when Vaaa does not equal Bbbb.



Operand Data Type	DL06 Range	
..... B	aaa	bbb
V memory .....	See memory map	See memory map
Pointer .....	See memory map	See memory map
Constant .....	—	0-9999

In the following example, when the value in V memory location V2000 = 4500 or V2002 ≠ 2500, Y3 will energize.

DirectSOFT32

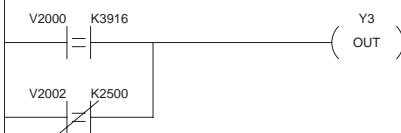


Handheld Programmer Keystrokes

\$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
E 4	F 5	A 0	A 0	ENT				
Q OR	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

In the following example, when the value in V memory location V2000 = 3916 or V2002 050, Y3 will energize.

DirectSOFT32

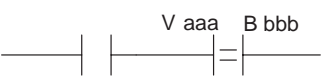


Handheld Programmer Keystrokes

\$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
D 3	J 9	B 1	G 6	ENT				
R ORN	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	D 3	ENT					

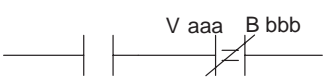
And If Equal (ANDE)

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



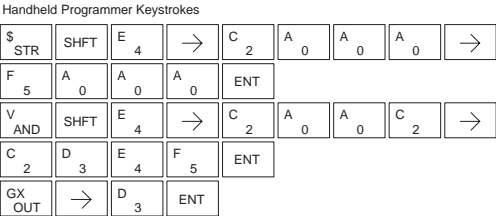
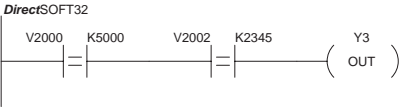
And If Not Equal (ANDNE)

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

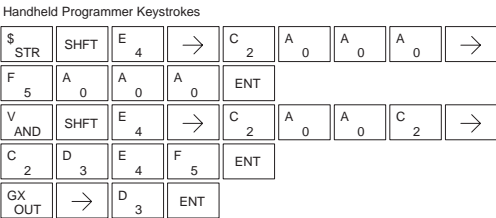
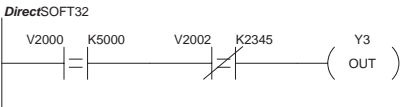


Operand Data Type	DL06 Range	
..... A/B	aaa	bbb
V memory .....	See memory map	See memory map
Pointer .....	See memory map	See memory map
Constant .....	—	0-9999

In the following example, when the value in V memory location V2000 = 5000 and V2002 = 2345, Y3 will energize.

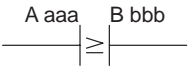


In the following example, when the value in V memory location V2000 = 5000 and V2002 ≠ 2345, Y3 will energize.



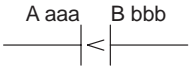
## Store (STR)

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



## Store Not (STRN)

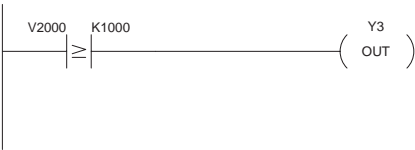
The Comparative Store Not instruction begins a new rung or additional branch in a rung with a normally closed comparative contact. The contact will be on when Aaaa < Bbbb.



Operand Data Type	DL06 Range	
A/B	aaa	bbb
V memory	See memory map	See memory map
Pointer	See memory map	See memory map
Constant	—	0–9999
Timer	0–377	
Counter	0–177	

In the following example, when the value in V memory location V2000 ≥ 1000, Y3 will energize.

DirectSOFT32



Handheld Programmer Keystrokes

\$	STR	→	SHFT	V	AND	C	2	A	0	A	0	A	0
→	B	1	A	0	A	0	A	0	ENT				
GX	OUT	→	D	3	ENT								

In the following example, when the value in V memory location V2000 < 4050, Y3 will energize.

DirectSOFT32



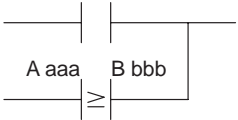
Handheld Programmer Keystrokes

SP	STRN	→	SHFT	V	AND	C	2	A	0	A	0	A	0
→	E	4	A	0	F	5	A	0	ENT				
GX	OUT	→	D	3	ENT								



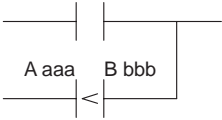
Or (OR)

The Comparative Or instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa is equal to or greater than Bbbb.



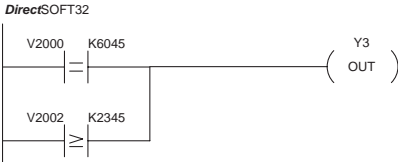
Or Not (ORN)

The Comparative Or Not instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa < Bbbb.

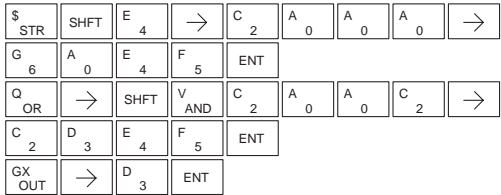


Operand Data Type	DL06 Range	
A/B	aaa	bbb
V memory	See memory map	See memory map
Pointer	See memory map	See memory map
Constant	—	0-9999
Timer	0-377	
Counter	0-177	

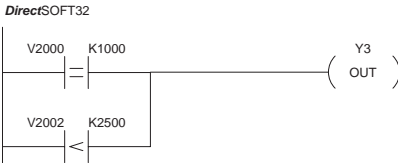
In the following example, when the value in V memory location V2000 = 6045 or V2002 ≥ 2345, Y3 will energize.



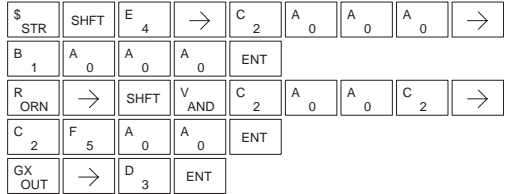
Handheld Programmer Keystrokes



In the following example when the value in V memory location V2000 = 1000 or V2002 < 2500, Y3 will energize.

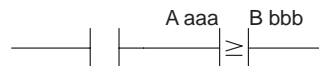


Handheld Programmer Keystrokes



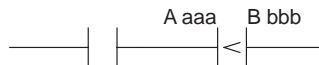
## And (AND)

The Comparative And instruction connects a normally open comparative contact in series with another contact. The contact will be on when Aaaa is equal to or greater than Bbbb.



## And Not (ANDN)

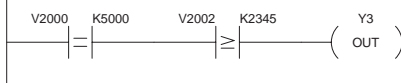
The Comparative And Not instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa < Bbbb.



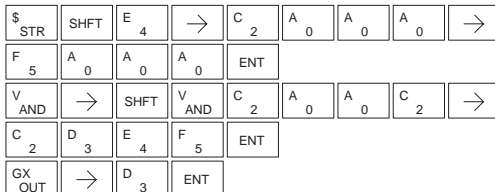
Operand Data Type	DL06 Range	
A/B	aaa	bbb
V memory	See memory map	See memory map
Pointer	See memory map	See memory map
Constant	—	0-9999
Timer	0-377	
Counter	0-177	

In the following example, when the value in V memory location V2000 = 5000, and  $V2002 \geq 2345$ , Y3 will energize.

DirectSOFT32

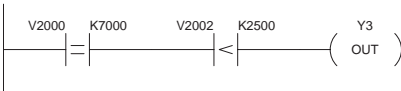


Handheld Programmer Keystrokes

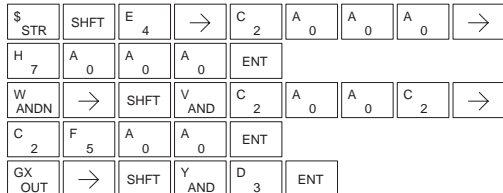


In the following example, when the value in V memory location V2000 = 7000 and  $V2002 < 2500$ , Y3 will energize.

DirectSOFT32



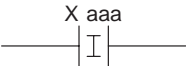
Handheld Programmer Keystrokes



# Immediate Instructions

## Store Immediate (STRI)

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



## Store Not Immediate (STRNI)

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



Operand Data Type	DL06 Range
	<b>aaa</b>
Inputs .....X	0-777

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

DirectSOFT32



Handheld Programmer Keystrokes

\$ STR	SHIFT	I 8	→	B 1	ENT
GX OUT	→	C 2	ENT		

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

DirectSOFT32

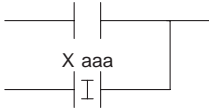


Handheld Programmer Keystrokes

SP STRN	SHIFT	I 8	→	B 1	ENT
GX OUT	→	C 2	ENT		

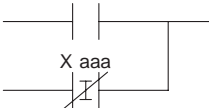
## Or Immediate (ORI)

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



## Or Not Immediate (ORNI)

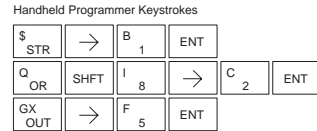
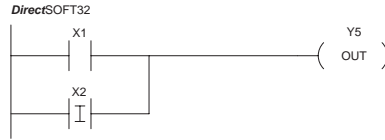
The Or Not Immediate connects two contacts in parallel. The status of the contact will be opposite the status of the associated input point *at the time the instruction is executed*. The image register is not updated.



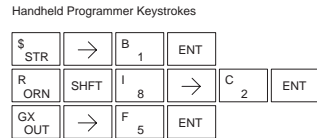
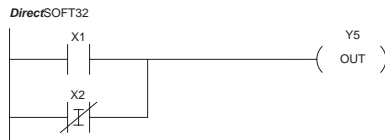
## OR Not Immediate Instructions Cont'd

Operand Data Type	DL06 Range
	<b>aaa</b>
Inputs ..... X	0-777

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

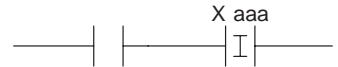


In the following example, when X1 is on or X2 is off, Y5 will energize.



## And Immediate (ANDI)

The And Immediate connects two contacts in series. The status of the contact will be the same as the status of the associated input point *at the time the instruction is executed*. The image register is not updated.



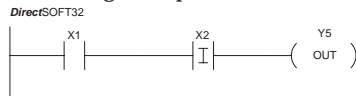
## And Not Immediate (ANDNI)

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

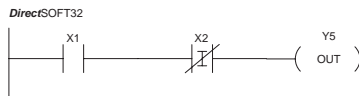


Operand Data Type	DL06 Range
	<b>aaa</b>
Inputs ..... X	0-777

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

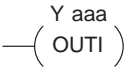


In the following example, when X1 is on and X2 is off, Y5 will energize.



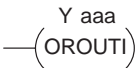
Out Immediate (OUTI)

The Out Immediate instruction reflects the status of the rung (on/off) and outputs the discrete (on/off) status to the specified module output point and the image register *at the time the instruction is executed*. If multiple Out Immediate instructions referencing the same discrete point are used it is possible for the module output status to change multiple times in a CPU scan. See Or Out Immediate.



Or Out Immediate (OROUTI)

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



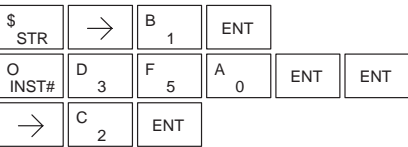
Operand Data Type	DL06 Range
	<b>aaa</b>
Outputs ..... Y	0-777

In the following example, when X1 is on, output point Y2 on the output module will turn on. For instruction entry on the Handheld Programmer, you can use the instruction number (#350) as shown, or type each letter of the command.

DirectSOFT32

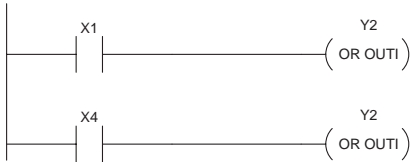


Handheld Programmer Keystrokes

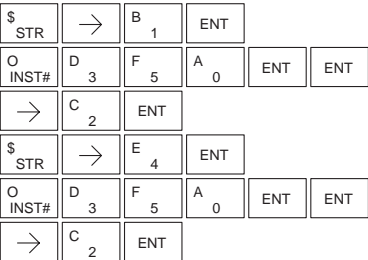


In the following example, when X1 or X4 is on, Y2 will energize.

DirectSOFT32



Handheld Programmer Keystrokes



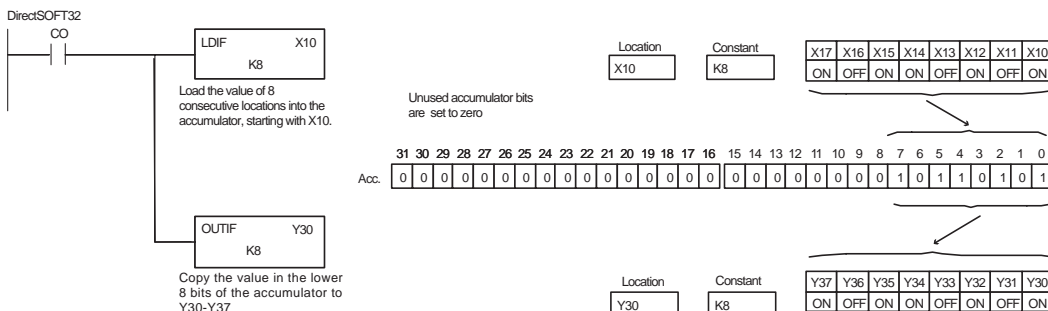
## Out Immediate Formatted (OUTIF)

The Out Immediate Formatted instruction outputs a 1–32 bit binary value from the accumulator to specified output points *at the time the instruction is executed*. Accumulator bits that are not used by the instruction are set to zero.

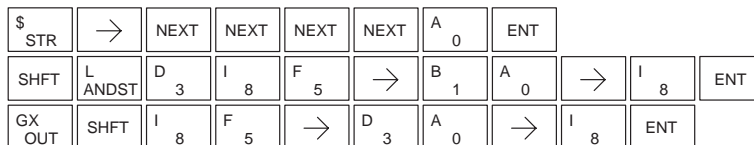


Operand Data Type	DL06 Range
	<b>aaa</b>
Outputs ..... Y	0-777
Constant ..... K	1-32

In the following example when C0 is on, the binary pattern for X10 –X17 is loaded into the accumulator using the Load Immediate Formatted instruction. The binary pattern in the accumulator is written to Y30–Y37 using the Out Immediate Formatted instruction. This technique is useful to quickly copy an input pattern to outputs (without waiting for the CPU scan).

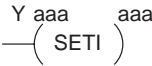


### Handheld Programmer Keystrokes



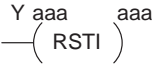
Set Immediate (SETI)

The Set Immediate instruction immediately sets, or turns on an output or a range of outputs in the image register and the corresponding output point(s) *at the time the instruction is executed*. Once the outputs are set it is not necessary for the input to remain on. The Reset Immediate instruction can be used to reset the outputs.



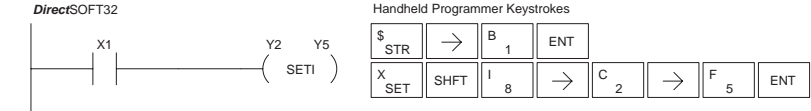
Reset Immediate (RSTI)

The Reset Immediate instruction immediately resets, or turns off an output or a range of outputs in the image register and the output point(s) *at the time the instruction is executed*. Once the outputs are reset it is not necessary for the input to remain on.

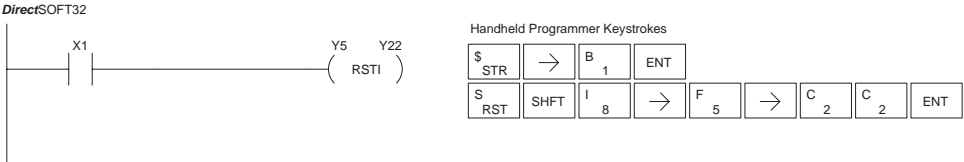


Operand Data Type	DL06 Range
	<b>aaa</b>
Outputs .....Y	0-777

In the following example, when X1 is on, Y2 through Y5 will be set on in the image register and on the corresponding output points.

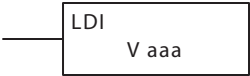


In the following example, when X1 is on, Y5 through Y22 will be reset (off) in the image register and on the corresponding output module(s).



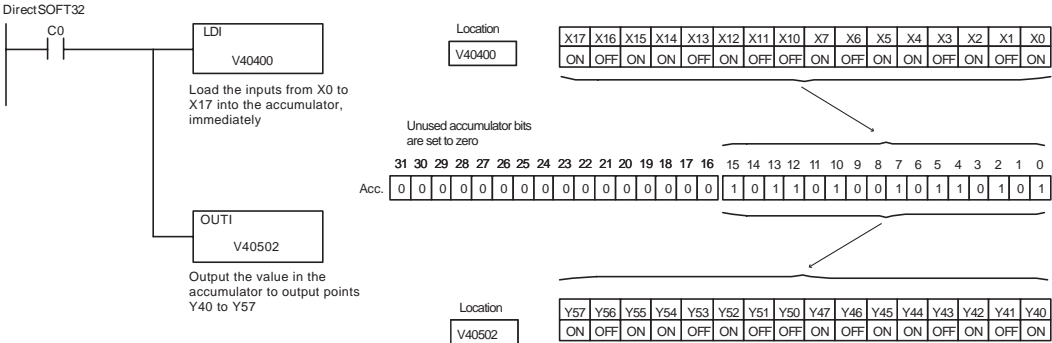
Load Immediate (LDI)

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



Operand Data Type	DL06 Range
	aaa
Inputs V	40400 - 40437

In the following example, when C0 is on, the binary pattern of X0–X17 will be loaded into the accumulator using the Load Immediate instruction. The Out Immediate instruction could be used to copy the 16 bits in the accumulator to output points, such as Y40–Y57. This technique is useful to quickly copy an input pattern to output points (without waiting for a full CPU scan to occur).



Handheld Programmer Keystrokes

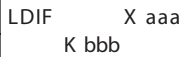
\$ STR	→	NEXT	NEXT	NEXT	NEXT	A 0	ENT									
SHFT	L ANDST	D 3	I 8	→	E 4	A 0	E 4	A 0	A 0	ENT						
GX OUT	SHFT	I 8	→	NEXT	E 4	A 0	F 5	A 0	C 2	ENT						



Load Immediate Formatted (LDIF)

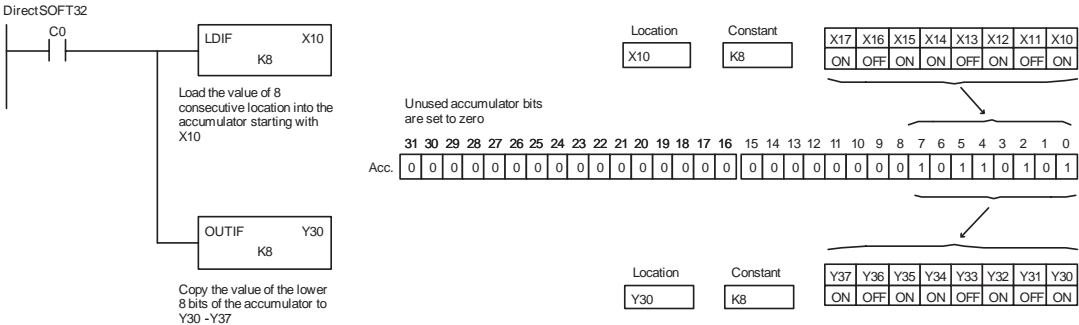
The Load Immediate Formatted instruction loads a 1–32 bit binary value into the accumulator. The value reflects the current status of the input module(s) *at the time the instruction is executed*.

Accumulator bits that are not used by the instruction are set to zero.

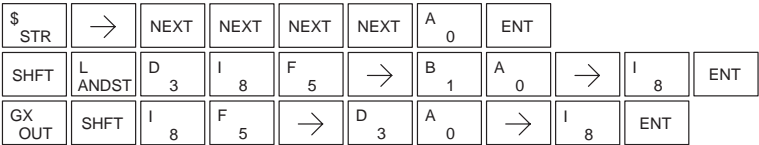


Operand Data Type	DL06 Range	
	aaa	bbb
Inputs ..... X	0-777	--
Constant ..... K	--	1-32

In the following example, when C0 is on, the binary pattern of X10–X17 will be loaded into the accumulator using the Load Immediate Formatted instruction. The Out Immediate Formatted instruction could be used to copy the specified number of bits in the accumulator to the specified outputs on the output module, such as Y30–Y37. This technique is useful to quickly copy an input pattern to outputs (without waiting for the CPU scan).



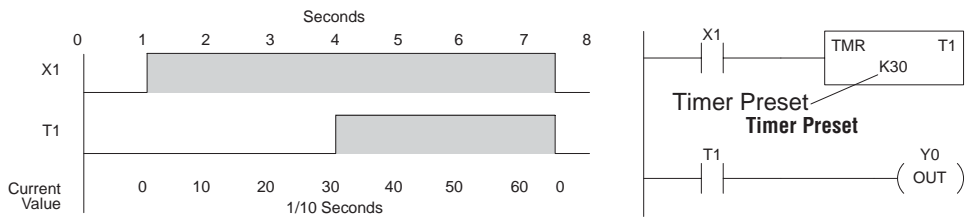
Handheld Programmer Keystrokes



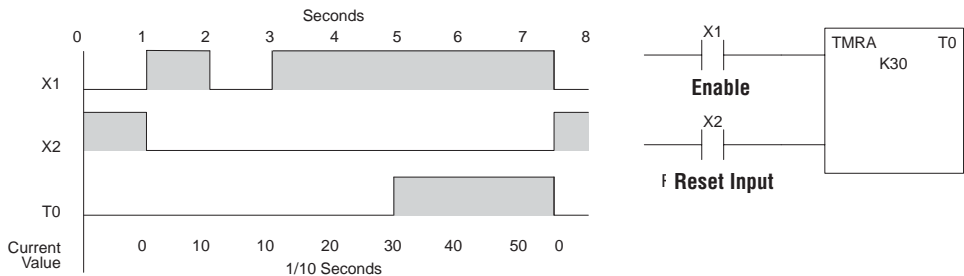
## Timer, Counter and Shift Register Instructions

### Using Timers

Timers are used to time an event for a desired length of time. The single input timer will time as long as the input is on. When the input changes from on to off the timer current value is reset to 0. There is a tenth of a second and a hundredth of a second timer available with a maximum time of 999.9 and 99.99 seconds respectively. There is a discrete bit associated with each timer to indicate that the current value is equal to or greater than the preset value. The timing diagram below shows the relationship between the timer input, associated discrete bit, current value, and timer preset.



There are those applications that need an accumulating timer, meaning it has the ability to time, stop, and then resume from where it previously stopped. The accumulating timer works similarly to the regular timer, but two inputs are required. The enable input starts and stops the timer. When the timer stops, the elapsed time is maintained. When the timer starts again, the timing continues from the elapsed time. When the reset input is turned on, the elapsed time is cleared and the timer will start at 0 when it is restarted. There is a tenth of a second and a hundredth of a second timer available with a maximum time of 999999.9 and 99999.99 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)

The Timer instruction is a 0.1 second single input timer that times to a maximum of 999.9 seconds. The Timer Fast instruction is a 0.01 second single input timer that times up to a maximum of 99.99 seconds. These timers will be enabled if the input logic is true (on) and will be reset to 0 if the input logic is false (off).

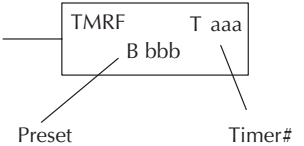
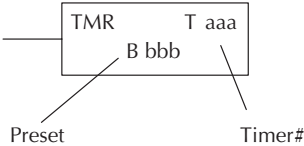
Instruction Specifications

Timer Reference (Taaa): Specifies the timer number.

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

Current Value: Timer current values are accessed by referencing the associated V or T memory location\*. For example, the timer current value for T3 physically resides in V-memory location V3.

Discrete Status Bit: The discrete status bit is referenced by the associated T memory location. Operating as a “timer done bit”, it will be on if the current value is equal to or greater than the preset value. For example, the discrete status bit for Timer 2 is T2.



**NOTE:** Timer preset constants (K) may be changed by using a handheld programmer, even when the CPU is in Run Mode. Therefore, a V-memory preset is required only if the ladder program must change the preset.

Operand Data Type	DL06 Range	
..... A/B	aaa	bbb
Timers ..... T	0-777	—
V memory for preset values ..... V	—	1200-7377 7400-7577 10000-17777
Pointers (preset only) ..... P	—	1200-7377 7400-7577 10000-17777
Constants (preset only) ..... K	—	0-9999
Timer discrete status bits ..... T/V	0-377 or V41100-41107	
Timer current values ..... V /T*	0-377	



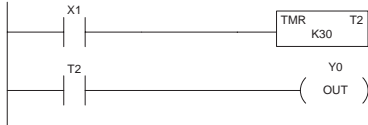
**NOTE:** \* With the HPP, both the Timer discrete status bits and current value are accessed with the same data reference. **DirectSOFT** uses separate references, such as “T2” for discrete status bit for Timer T2, and “TA2” for the current value of Timer T2.

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

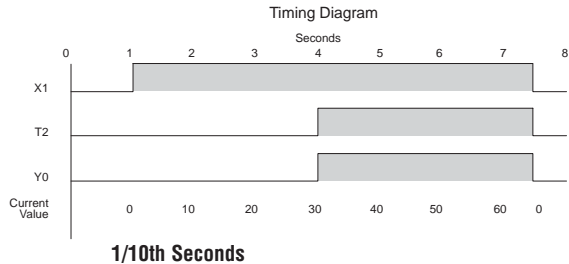
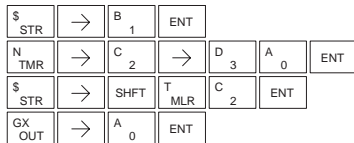
## Timer Example Using Discrete Status Bits

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

Direct SOFT32



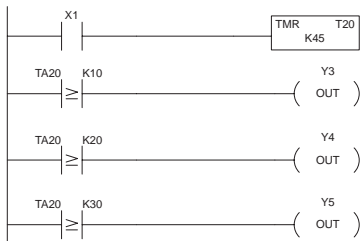
Handheld Programmer Keystrokes



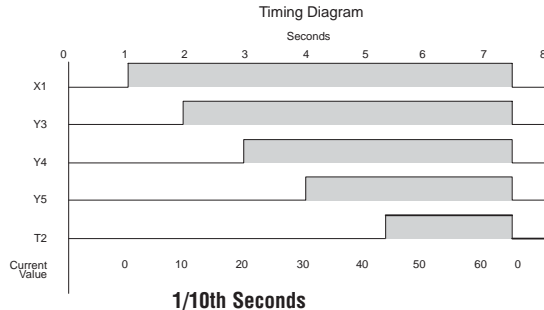
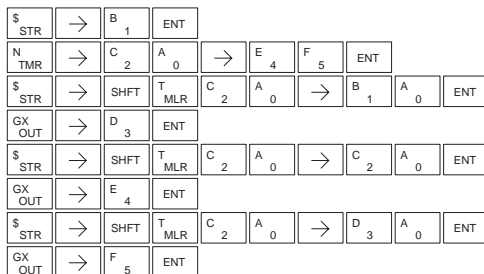
## Timer Example Using Comparative Contacts

In the following example, a single input timer is used with a preset of 4.5 seconds. Comparative contacts are used to energize Y3, Y4, and Y5 at one second intervals respectively. When X1 is turned off the timer will be reset to 0 and the comparative contacts will turn off Y3, Y4, and Y5.

Direct SOFT32



Handheld Programmer Keystrokes



## Accumulating Timer (TMRA)

The Accumulating Timer is a 0.1 second two input timer that will time to a maximum of 9999999.9.

## Accumulating Fast Timer (TMRAF)

The Accumulating Fast Timer is a 0.01 second two-input timer that will time to a maximum of 99999.99.

*Each one uses two timer registers in V-memory.* These timers have two inputs, an enable and a reset. The timer starts timing when the enable is on and stops when the enable is off (without resetting the count). The reset will reset the timer when on and allow the timer to time when off.

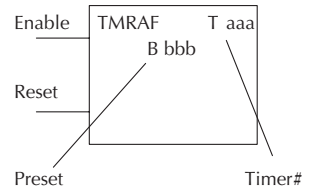
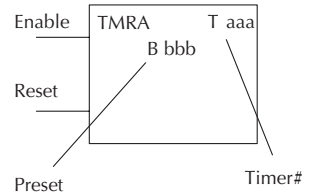
### Instruction Specifications

**Timer Reference (Taaa):** Specifies the timer number.

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

**Current Value:** Timer current values are accessed by referencing the associated V or T memory location\*. For example, the timer current value for T3 resides in V-memory location V3.

**Discrete Status Bit:** The discrete status bit is accessed by referencing the associated T memory location. Operating as a “timer done bit,” it will be on if the current value is equal to or greater than the preset value. For example the discrete status bit for timer 2 would be T2.



**NOTE:** The accumulating type timer uses **two consecutive V-memory locations** for the 8-digit value, and therefore two consecutive timer locations. For example, if TMRA 1 is used, the next available timer number is TMRA 3.

Operand Data Type	DL06 Range	
A/B	aaa	bbb
Timers ..... T	0-376	—
V memory for preset values ..... V	—	1200-7377 7400-7577 10000-17777
Pointers (preset only) ..... P	—	1200-7377 7400-7577 10000-17777
Constants (preset only) ..... K	—	0-99999999
Timer discrete status bits ..... T/V	0-376 or V41100-41117	
Timer current values ..... V / T*	0-376	



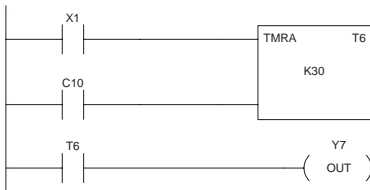
**NOTE:** \* With the HPP, both the Timer discrete status bits and current value are accessed with the same data reference. **DirectSOFT** uses separate references, such as “T2” for discrete status bit for Timer T2, and “TA2” for the current value of Timer T2.

The following examples show two methods of programming timers. One performs functions when the timer reaches the preset value using the discrete status bit, or use comparative contacts to perform functions at different time intervals.

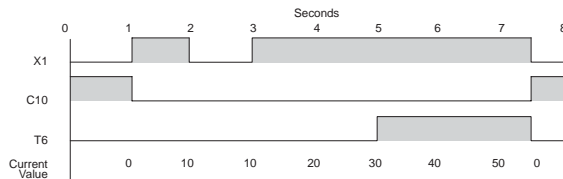
## Accumulating Timer Example using Discrete Status Bits

In the following example, a two input timer (accumulating timer) is used with a preset of 3 seconds. The timer discrete status bit (T6) will turn on when the timer has timed for 3 seconds. Notice in this example that the timer times for 1 second, stops for one second, then resumes timing. The timer will reset when C10 turns on, turning the discrete status bit off and resetting the timer current value to 0.

Direct SOFT32



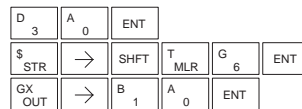
Timing Diagram



Handheld Programmer Keystrokes



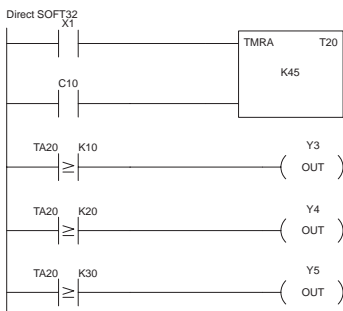
Handheld Programmer Keystrokes (cont)



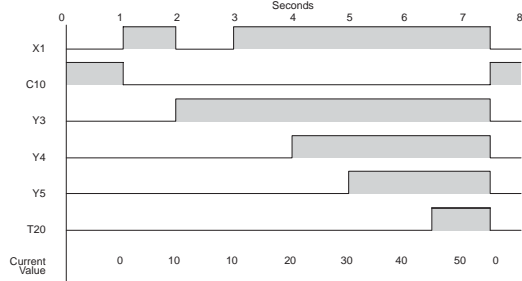
## Accumulator Timer Example Using Comparative Contacts

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

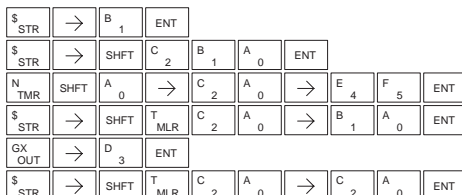
Contacts



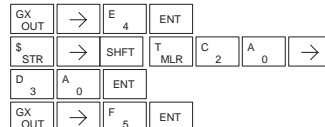
Timing Diagram



Handheld Programmer Keystrokes



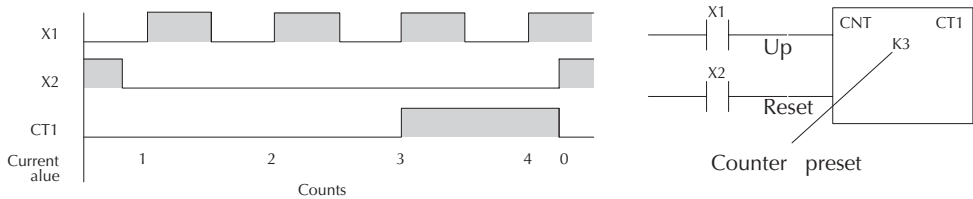
Handheld Programmer Keystrokes (cont)



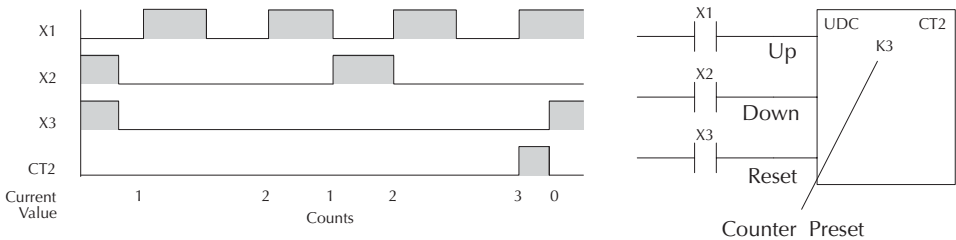
### Using Counters

Counters are used to count events. The counters available are up counters, up/down counters, and stage counters (used with RLL<sup>PLUS</sup> programming).

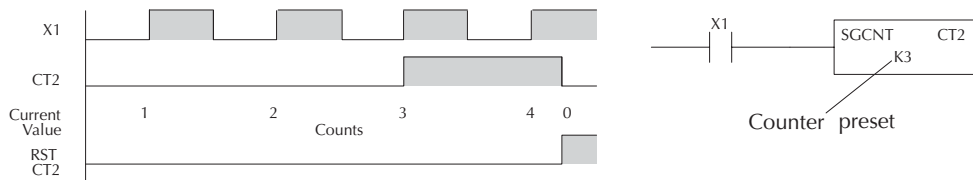
The up counter (CNT) has two inputs, a count input and a reset input. The maximum count value is 9999. The timing diagram below shows the relationship between the counter input, counter reset, associated discrete bit, current value, and counter preset.



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



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



# Counter (CNT)

The Counter is a two input counter that increments when the count input logic transitions from off to on. When the counter reset input is on the counter resets to 0. When the current value equals the preset value, the counter status bit comes on and the counter continues to count up to a maximum count of 9999. The maximum value will be held until the counter is reset.

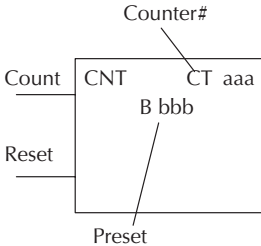
## Instruction Specifications

**Counter Reference (CTaaa):** Specifies the counter number.

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

**Current Values:** Counter current values are accessed by referencing the associated V or CT memory locations\*. The V-memory location is the counter location + 1000. For example, the counter current value for CT3 resides in V memory location V1003.

**Discrete Status Bit:** The discrete status bit is accessed by referencing the associated CT memory location. It will be on if the value is equal to or greater than the preset value. For example the discrete status bit for counter 2 would be CT2.



**NOTE:** Counter preset constants (K) may be changed by using a programming device, even when the CPU is in Run Mode. Therefore, a V-memory preset is required only if the ladder program must change the preset.

Operand Data Type	DL06 Range	
	aaa	bbb
Counters ..... A/B	aaa	bbb
Counters ..... CT	0-177	—
V memory (preset only) ..... V	—	1200-7377 7400-7577 10000-17777
Pointers (preset only) ..... P	—	1200-7377 7400-7577 10000-17777
Constants (preset only) ..... K	—	0-9999
Counter discrete status bits ..... CT/V	0-177 or V41140-41147	
Counter current values ..... V/CT*	1000-1177	



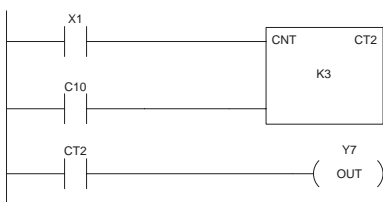
**NOTE:** \* With the HPP, both the Counter discrete status bits and current value are accessed with the same data reference. **DirectSOFT** uses separate references, such as “CT2” for discrete status bit for Counter CT2, and “CTA2” for the current value of Counter CT2.



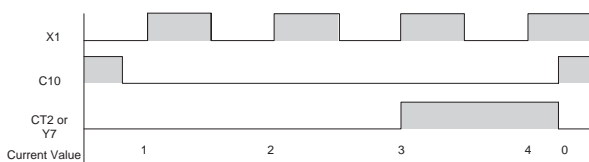
## Counter Example Using Discrete Status Bits

In the following example, when X1 makes an off to on transition, counter CT2 will increment by one. When the current value reaches the preset value of 3, the counter status bit CT2 will turn on and energize Y7. When the reset C10 turns on, the counter status bit will turn off and the current value will be 0. The current value for counter CT2 will be held in V memory location V1002.

DirectSOFT32



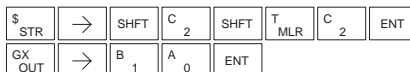
Counting diagram



Handheld Programmer Keystrokes



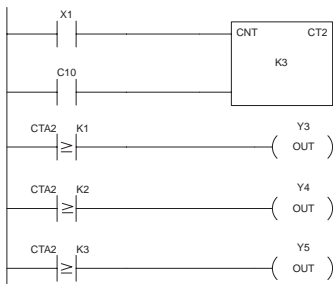
Handheld Programmer Keystrokes (cont)



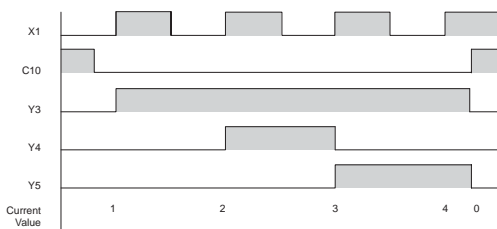
## Counter Example Using Comparative Contacts

In the following example, when X1 makes an off to on transition, counter CT2 will increment by one. Comparative contacts are used to energize Y3, Y4, and Y5 at different counts. When the reset C10 turns on, the counter status bit will turn off and the counter current value will be 0, and the comparative contacts will turn off.

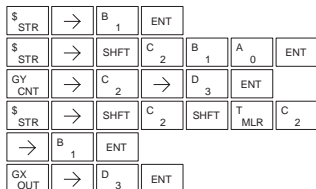
DirectSOFT32



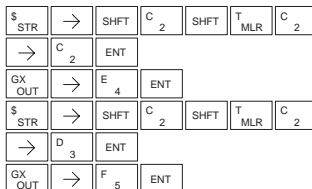
Counting diagram



Handheld Programmer Keystrokes

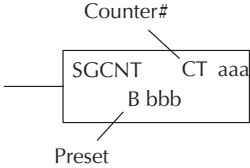


Handheld Programmer Keystrokes (cont)



# Stage Counter (SGCNT)

The Stage Counter is a single input counter that increments when the input logic transitions from off to on. This counter differs from other counters since it will hold its current value until reset using the RST instruction. The Stage Counter is designed for use in RLL<sup>PLUS</sup> programs but can be used in relay ladder logic programs. When the current value equals the preset value, the counter status bit turns on and the counter continues to count up to a maximum count of 9999. The maximum value will be held until the counter is reset.



## Instruction Specifications

**Counter Reference (CTaaa):** Specifies the counter number.

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

**Current Values:** Counter current values are accessed by referencing the associated V or CT memory locations\*. The V-memory location is the counter location + 1000. For example, the counter current value for CT3 resides in V memory location V1003.

**Discrete Status Bit:** The discrete status bit is accessed by referencing the associated CT memory location. It will be on if the value is equal to or greater than the preset value. For example the discrete status bit for counter 2 would be CT2.

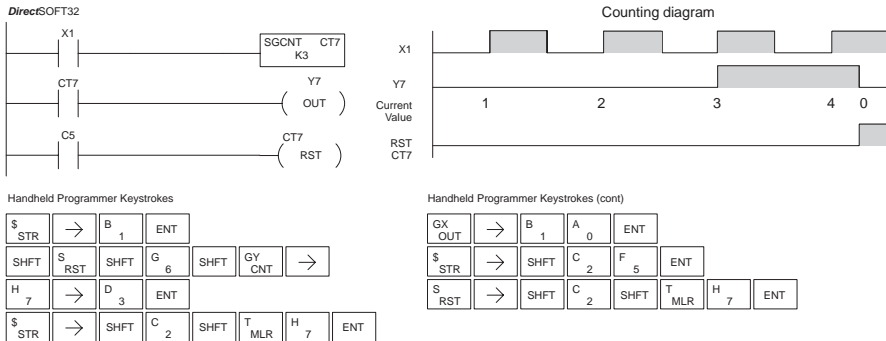
Operand Data Type	DL06 Range	
..... A/B	aaa	bbb
Counters ..... CT	0-177	—
V memory (preset only) ..... V	—	1200-7377 7400-7577 10000-17777
Pointers (preset only) ..... P	—	1200-7377 7400-7577 10000-17777
Constants (preset only) ..... K	—	0-9999
Counter discrete status bits ..... CT/V	0-177 or V41140-41147	
Counter current values ..... V /CT*	1000-1177	



**NOTE:** \* With the HPP, both the Counter discrete status bits and current value are accessed with the same data reference. **DirectSOFT** uses separate references, such as “CT2” for discrete status bit for Counter CT2, and “CTA2” for the current value of Counter CT2.

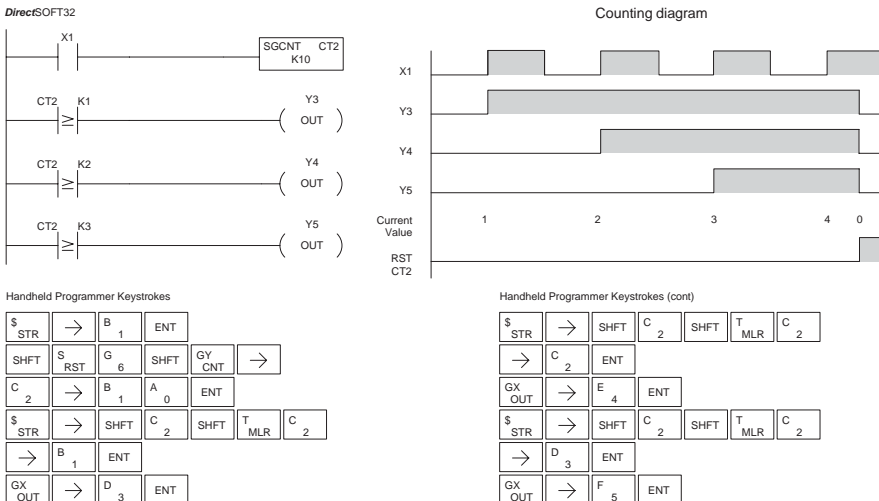
## Stage Counter Example Using Discrete Status Bits

In the following example, when X1 makes an off to on transition, stage counter CT7 will increment by one. When the current value reaches 3, the counter status bit CT7 will turn on and energize Y7. The counter status bit CT7 will remain on until the counter is reset using the RST instruction. When the counter is reset, the counter status bit will turn off and the counter current value will be 0. The current value for counter CT7 will be held in V memory location V1007.



## Stage Counter Example Using Comparative Contacts

In the following example, when X1 makes an off to on transition, counter CT2 will increment by one. Comparative contacts are used to energize Y3, Y4, and Y5 at different counts. Although this is not shown in the example, when the counter is reset using the Reset instruction, the counter status bit will turn off and the current value will be 0. The current value for counter CT2 will be held in V memory location V1002.



# Up Down Counter (UDC)

This Up/Down Counter counts up on each off to on transition of the Up input and counts down on each off to on transition of the Down input. The counter is reset to 0 when the Reset input is on. The count range is 0–99999999. The count input not being used must be off in order for the active count input to function.

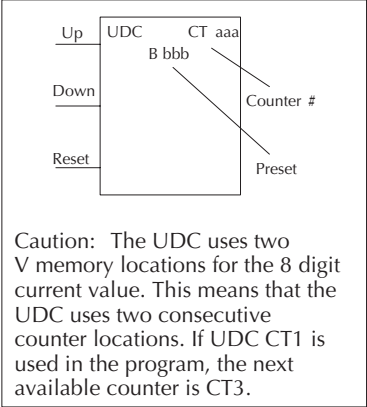
## Instruction Specification

**Counter Reference (CTaaa):** Specifies the counter number.

**Preset Value (Bbbb):** Constant value (K) or two consecutive V memory locations.

**Current Values:** Current count is a double word value accessed by referencing the associated V or CT memory locations\*. The V-memory location is the counter location + 1000. For example, the counter current value for CT5 resides in V memory location V1005 and V1006.

**Discrete Status Bit:** The discrete status bit is accessed by referencing the associated CT memory location. Operating as a “counter done bit” it will be on if the value is equal to or greater than the preset value. For example the discrete status bit for counter 2 would be CT2.



**The counter discrete status bit and the current value are not specified in the counter instruction**

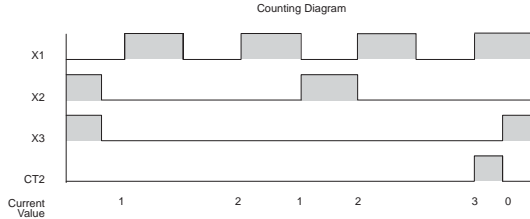
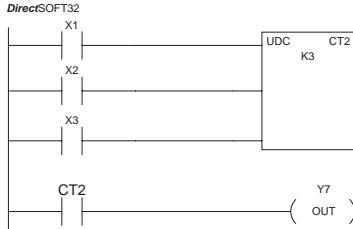
Operand Data Type	DL06 Range	
..... A/B	aaa	bbb
Counters ..... CT	0–176	—
V memory (preset only) ..... V	—	1200–7377 7400–7577 10000–17777
Pointers (preset only) ..... P	—	1200–7377 7400–7577 10000–17777
Constants (preset only) ..... K	—	0–99999999
Counter discrete status bits ..... CT/V	0–176 or V41140–41147	
Counter current values ..... V /CT*	1000–1176	



**NOTE:** \* With the HPP, both the Counter discrete status bits and current value are accessed with the same data reference. **DirectSOFT32** uses separate references, such as “CT2” for discrete status bit for Counter CT2, and “CTA2” for the current value of Counter CT2.

## Up / Down Counter Example Using Discrete Status Bits

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



Handheld Programmer Keystrokes

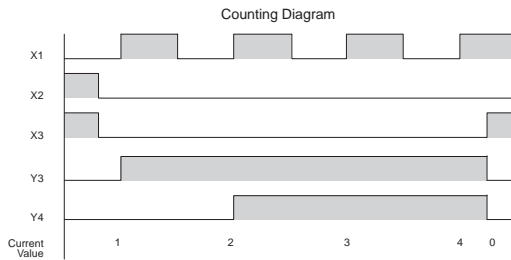
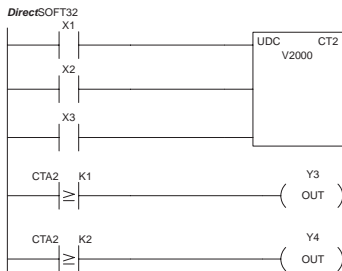
\$ STR	→	B 1	ENT
\$ STR	→	C 2	ENT
\$ STR	→	D 3	ENT
SHFT	U ISG	D 3	C 2
	→	C 2	

Handheld Programmer Keystrokes (cont)

→	D 3	ENT
\$ STR	→	SHFT C 2
		SHFT T MLR C 2
GX OUT	→	B 1 A 0
		ENT

## Up / Down Counter Example Using Comparative Contacts

In the following example, when X1 makes an off to on transition, counter CT2 will increment by one. Comparative contacts are used to energize Y3 and Y4 at different counts. When the reset (X3) turns on, the counter status bit will turn off, the current value will be 0, and the comparative contacts will turn off.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
\$ STR	→	C 2	ENT
\$ STR	→	D 3	ENT
SHFT	U ISG	D 3	C 2
	→	C 2	→
SHFT	V AND	C 2	A 0
		A 0	A 0
\$ STR	→	SHFT C 2	SHFT T MLR C 2

Handheld Programmer Keystrokes (cont)

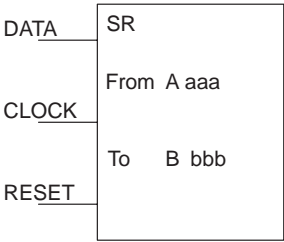
→	B 1	ENT
GX OUT	→	D 3
		ENT
\$ STR	→	SHFT C 2
		SHFT T MLR C 2
→	C 2	ENT
GX OUT	→	E 4
		ENT

# Shift Register (SR)

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 use 8-bit blocks.

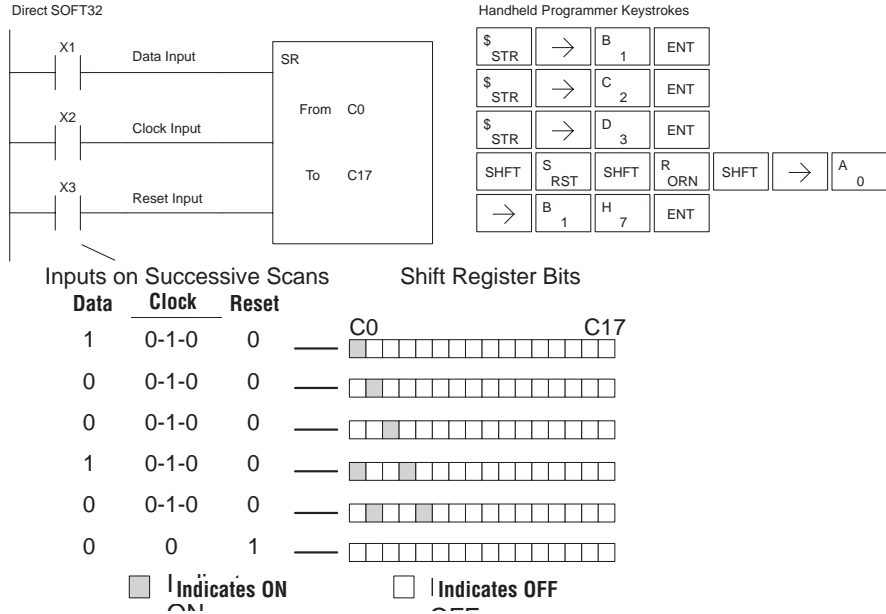
The Shift Register has three contacts.

- Data — determines the value (1 or 0) that will enter the register
- Clock — shifts the bits one position on each low to high transition
- Reset — resets the Shift Register to all zeros.



With each off to on transition of the clock input, the bits which make up the shift register block are shifted by one bit position and the status of the data input is placed into the starting bit position in the shift register. The direction of the shift depends on the entry in the From and To fields. From C0 to C17 would define a block of sixteen bits to be shifted from left to right. From C17 to C0 would define a block of sixteen bits, to be shifted from right to left. The maximum size of the shift register block depends on the number of available control relays. The minimum block size is 8 control relays.

Operand Data Type	DL06 Range	
..... A/B	aaa	bbb
Control Relay ..... C	0-1777	0-1777



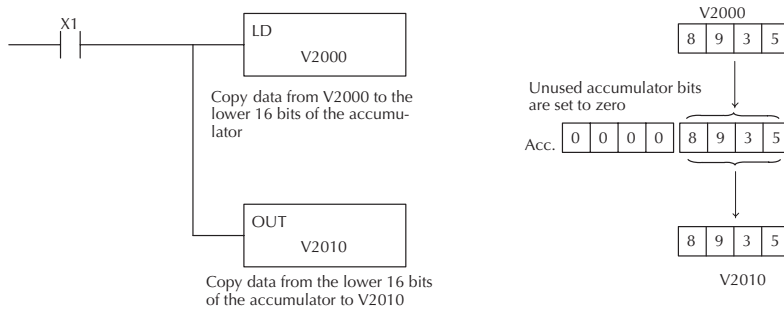
## Accumulator / Stack Load and Output Data Instructions

### Using the Accumulator

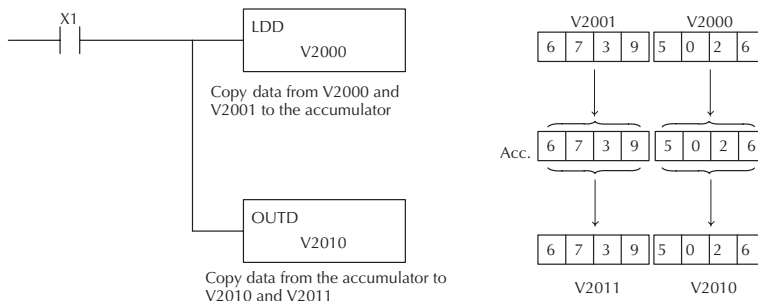
The accumulator in the DL06 internal CPUs is a 32 bit register which is used as a temporary storage location for data that is being copied or manipulated in some manner. For example, you have to use the accumulator to perform math operations such as add, subtract, multiply, etc. Since there are 32 bits, you can use up to an 8-digit BCD number. The accumulator is reset to 0 at the end of every CPU scan.

### Copying Data to the Accumulator

The Load and Out instructions and their variations are used to copy data from a V-memory location to the accumulator, or to copy data from the accumulator to V memory. The following example copies data from V-memory location V2000 to V-memory location V2010.

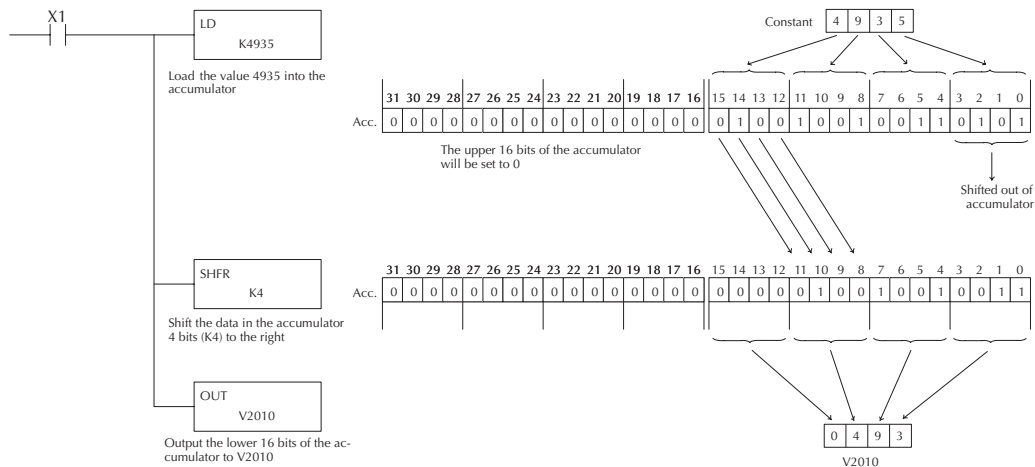


Since the accumulator is 32 bits and V memory locations are 16 bits, the Load Double and Out Double (or variations thereof) use two consecutive V-memory locations or 8 digit BCD constants to copy data either to the accumulator from a V-memory address or from a V-memory address to the accumulator. For example if you wanted to copy data from V2000 and V2001 to V2010 and V2011 the most efficient way to perform this function would be as follows:



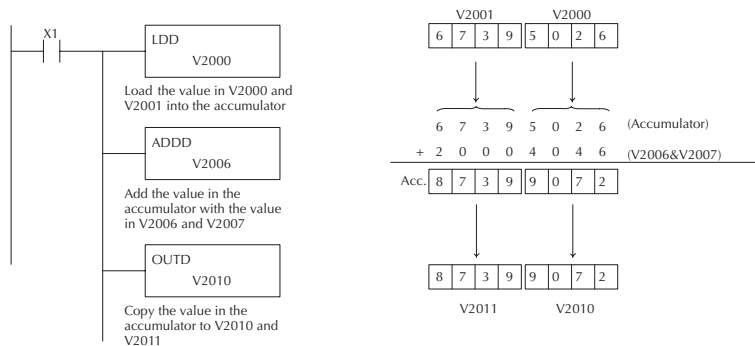
## Changing the Accumulator Data

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 value 4935 into the accumulator, shifts the data right 4 bits, and outputs the result to V2010.



Some of the data manipulation instructions use 32 bits. They use two consecutive V memory locations or an 8 digit BCD constant to manipulate data in the accumulator.

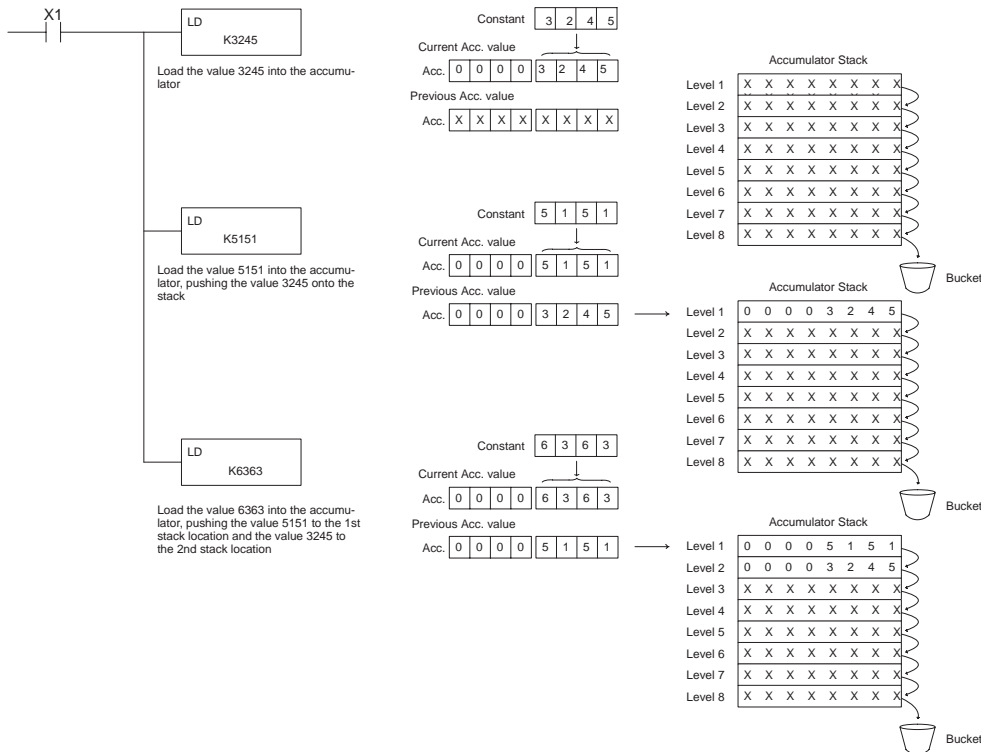
In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is added with the value in V2006 and V2007 using the Add Double instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



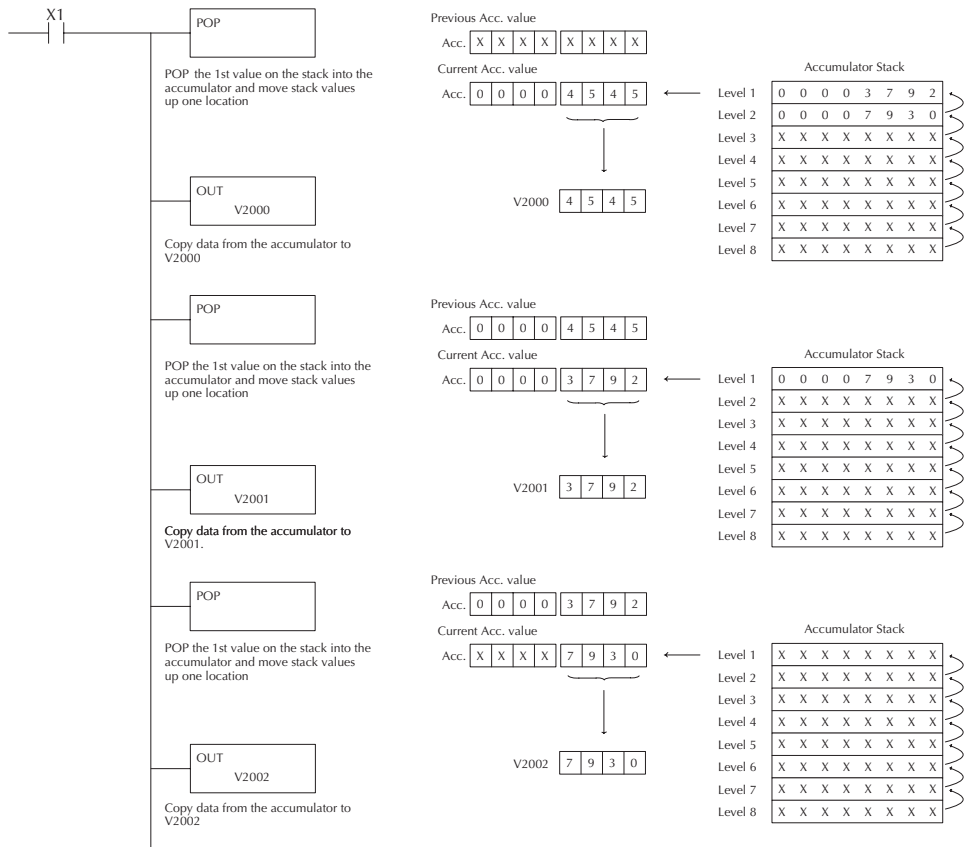


## Using the Accumulator Stack

The accumulator stack is used for instructions that require more than one parameter to execute a function or for user defined functionality. The accumulator stack is used when more than one Load instruction is executed without the use of an Out instruction. The first load instruction in the scan places a value into the accumulator. Every Load instruction thereafter without the use of an Out instruction places a value into the accumulator and the value that was in the accumulator is placed onto the accumulator stack. The Out instruction nullifies the previous load instruction and does not place the value that was in the accumulator onto the accumulator stack when the next load instruction is executed. Every time a value is placed onto the accumulator stack the other values in the stack are pushed down one location. The accumulator is eight levels deep (eight 32 bit registers). If there is a value in the eighth location when a new value is placed onto the stack, the value in the eighth location is pushed off the stack and cannot be recovered.



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



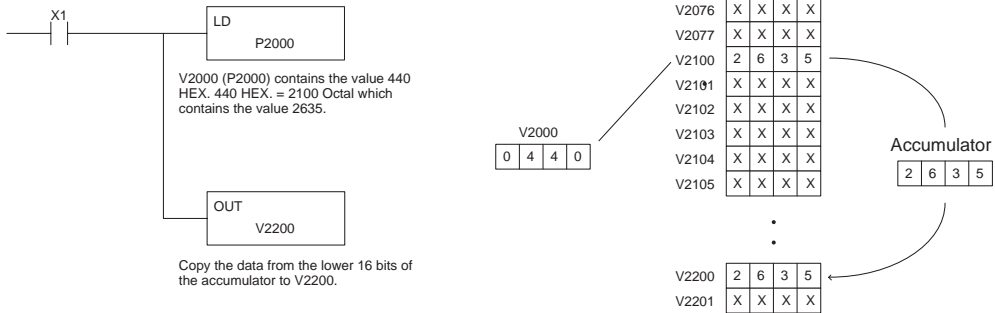
## Using Pointers

Many of the DL06 series instructions will allow V-memory pointers as an operand (commonly known as indirect addressing). Pointers allow instructions to obtain data from V-memory locations referenced by the pointer value.

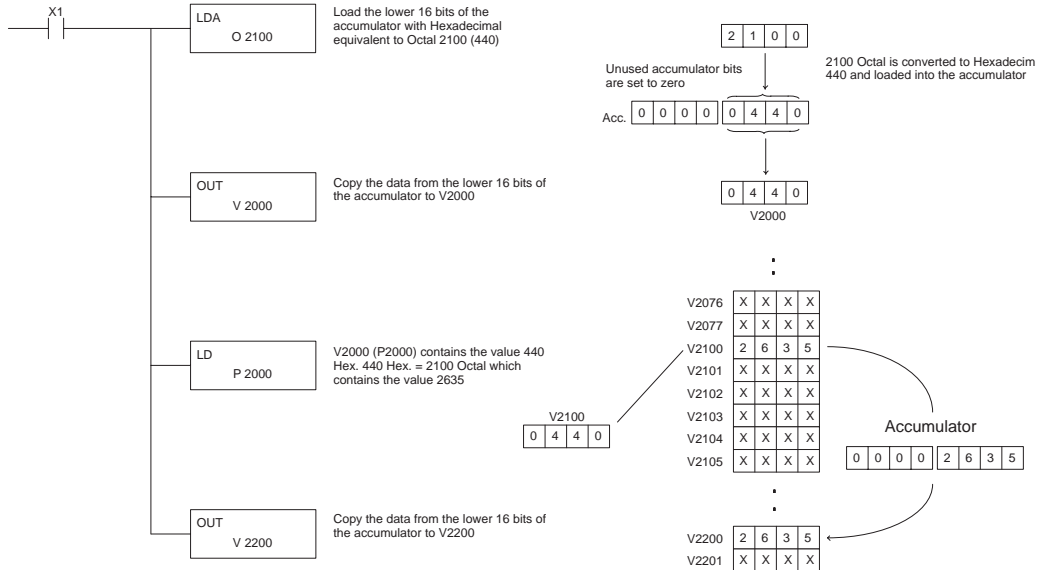


**NOTE:** DL06 V-memory addressing is in octal. However, the pointers reference a V-memory location with values viewed as HEX. Use the Load Address (LDA) instruction to move an address into the pointer location. This instruction performs the Octal to Hexadecimal conversion automatically.

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



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



## Load (LD)

The Load instruction is a 16 bit instruction that loads the value (Aaaa), which is either a V memory location or a 4 digit constant, into the lower 16 bits of the accumulator. The upper 16 bits of the accumulator are set to 0.



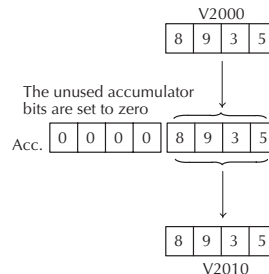
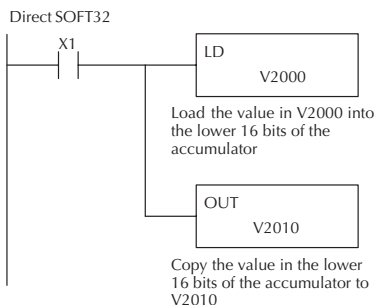
Operand Data Type	DL06 Range
..... A	<b>aaa</b>
V memory .....	See memory map
Pointer .....	See memory map
Constant .....	0-FFFF

Discrete Bit Flags	Description
SP53	on when the pointer is outside of the available range.
SP70	On anytime the value in the accumulator is negative.
SP76	On when any instruction loads a value of zero into the accumulator.

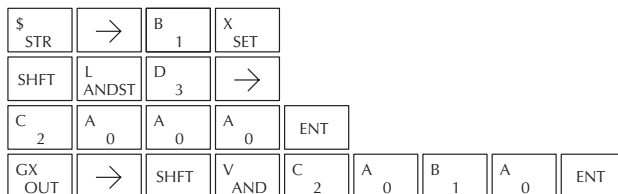


**NOTE:** Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.

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



Handheld Programmer Keystrokes



Load Double (LDD)

The Load Double instruction is a 32 bit instruction that loads the value (Aaaa), which is either two consecutive V memory locations or an 8 digit constant value, into the accumulator.

LDD  
A aaa

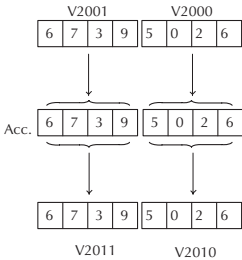
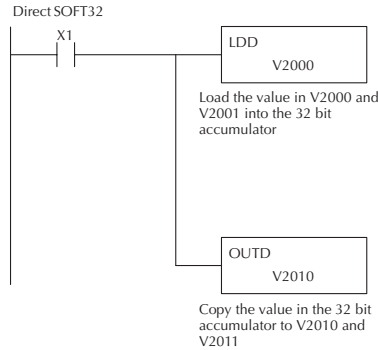
Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0-FFFF

Discrete Bit Flags	Description
SP53	on when the pointer is outside of the available range.
SP70	On anytime the value in the accumulator is negative.
SP76	On when any instruction loads a value of zero into the accumulator.



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

In the following example, when X1 is on, the 32 bit value in V2000 and V2001 will be loaded into the accumulator and output to V2010 and V2011.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
SHFT	L ANDST	D 3	D 3
C 2	A 0	A 0	A 0
GX OUT	SHFT	D 3	→
C 2	A 0	B 1	A 0

## Load Formatted (LDF)

The Load Formatted instruction loads 1–32 consecutive bits from discrete memory locations into the accumulator. The instruction requires a starting location (Aaaa) and the number of bits (Kbbb) to be loaded. Unused accumulator bit locations are set to zero.



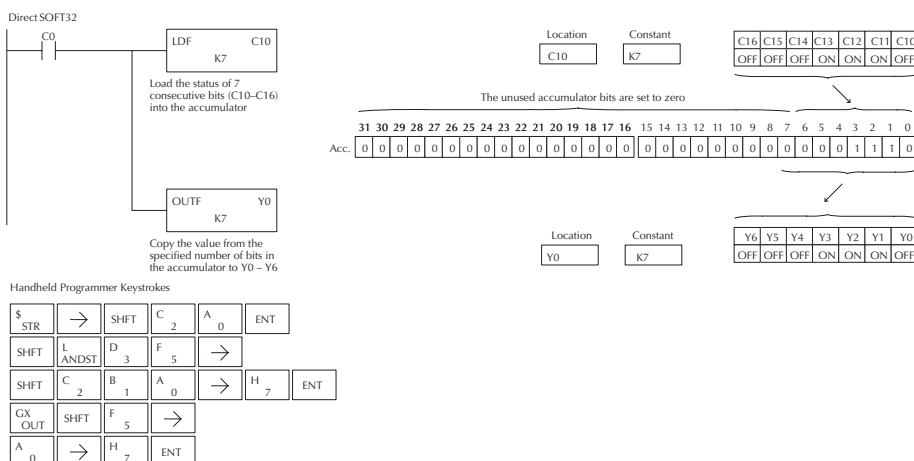
Operand Data Type		DL06 Range	
	A	aaa	bbb
Inputs	X	0–777	—
Outputs	Y	0–777	—
Control Relays	C	0–1777	—
Stage Bits	S	0–1777	—
Timer Bits	T	0–377	—
Counter Bits	CT	0–177	—
Special Relays	SP	0–777	—
Constant	K	—	1–32

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



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

In the following example, when C0 is on, the binary pattern of C10–C16 (7 bits) will be loaded into the accumulator using the Load Formatted instruction. The lower 7 bits of the accumulator are output to Y0–Y6 using the Out Formatted instruction.



Load Address (LDA)

The Load Address instruction is a 16 bit instruction. It converts any octal value or address to the HEX equivalent value and loads the HEX value into the accumulator. This instruction is useful when an address parameter is required since all addresses for the DL06 system are in octal.



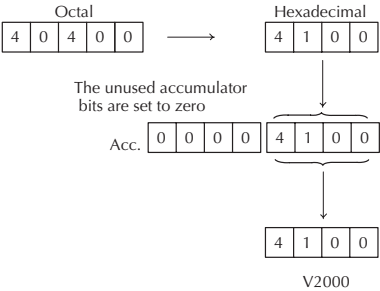
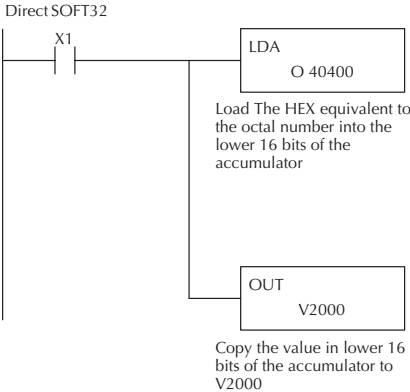
Operand Data Type	DL06 Range
	<b>aaa</b>
Octal Address ..... 0	See memory map

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

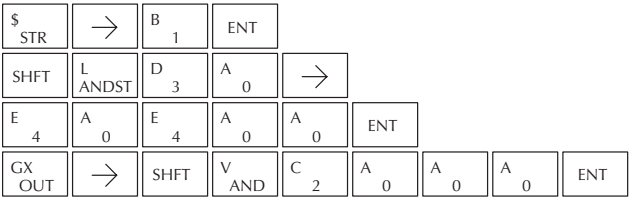


**NOTE:** Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.

In the following example when X1 is on, the octal number 40400 will be converted to a HEX 4100 and loaded into the accumulator using the Load Address instruction. The value in the lower 16 bits of the accumulator is copied to V2000 using the Out instruction.



Handheld Programmer Keystrokes







Load Accumulator Indexed from Data Constants (LDSX)

The Load Accumulator Indexed from Data Constants is a 16 bit instruction. The instruction specifies a Data Label Area (DLBL) where numerical or ASCII constants are stored. This value will be loaded into the lower 16 bits.



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

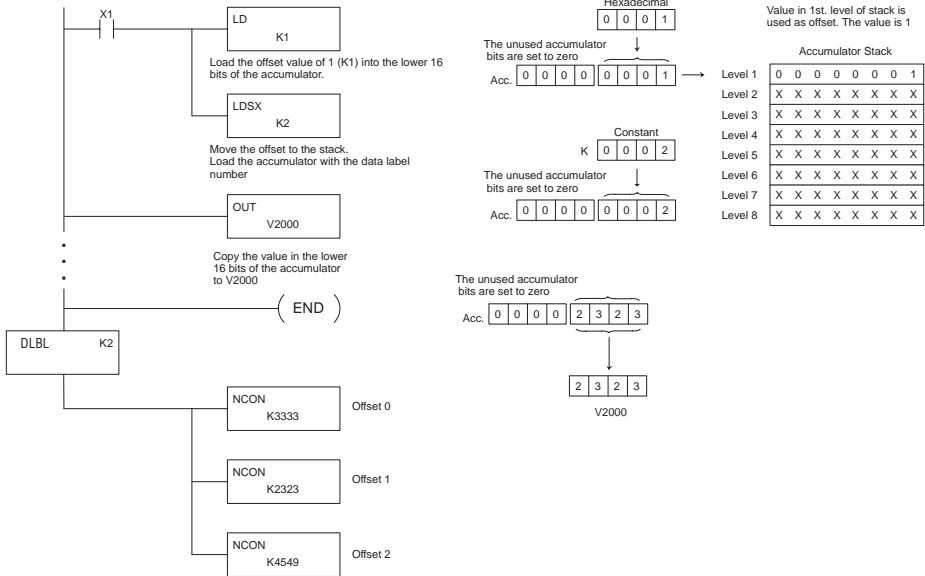
**Helpful Hint:** — The Load Address instruction can be used to convert octal to HEX and load the value into the accumulator.

Operand Data Type	DL06 Range
	<b>aaa</b>
Constant ..... K	1-FFFF



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

In the following example when X1 is on, the offset of 1 is loaded into the accumulator. This value will be placed into the first level of the accumulator stack when the LDSX instruction is executed. The LDSX instruction specifies the Data Label (DLBL K2) where the numerical constant(s) are located in the program and loads the constant value, indicated by the offset in the stack, into the lower 16 bits of the accumulator.



\$	STR	→	B <sub>1</sub>	ENT	Handheld Programmer Keystrokes													
SHFT	L ANDST	D <sub>3</sub>	→	SHFT	K JMP	B <sub>1</sub>	ENT											
SHFT	L ANDST	D <sub>3</sub>	S RST	X SET	→	C <sub>2</sub>	ENT											
SHFT	E <sub>4</sub>	N TMR	D <sub>3</sub>	ENT														
SHFT	D <sub>3</sub>	L ANDST	B <sub>1</sub>	L ANDST	→	C <sub>2</sub>	ENT											
SHFT	N TMR	C <sub>2</sub>	O INST#	N TMR	→	D <sub>3</sub>	D <sub>3</sub>	D <sub>3</sub>	D <sub>3</sub>	D <sub>3</sub>	ENT							
SHFT	N TMR	C <sub>2</sub>	O INST#	N TMR	→	C <sub>2</sub>	D <sub>3</sub>	C <sub>2</sub>	D <sub>3</sub>	C <sub>2</sub>	ENT							
SHFT	N TMR	C <sub>2</sub>	O INST#	N TMR	→	E <sub>4</sub>	F <sub>5</sub>	E <sub>4</sub>	J <sub>9</sub>	ENT								
GX_OUT	→	SHFT	V AND	C <sub>2</sub>	A <sub>0</sub>	A <sub>0</sub>	A <sub>0</sub>	ENT										

## Load Real Number (LDR)

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

LDR
A aaa

Operand Data Type	DL06 Range
.....A	<b>aaa</b>
V memory .....V	See memory map
Pointer .....P	See memory map
Real Constant .....R	-3.402823E+038 to +3.402823E+038

**Direct**SOFT32 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”.

LDR
R3.14159

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.

LDR
R5.3E6
OUTD
V1400

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.

LDR
V1400

Out (OUT)

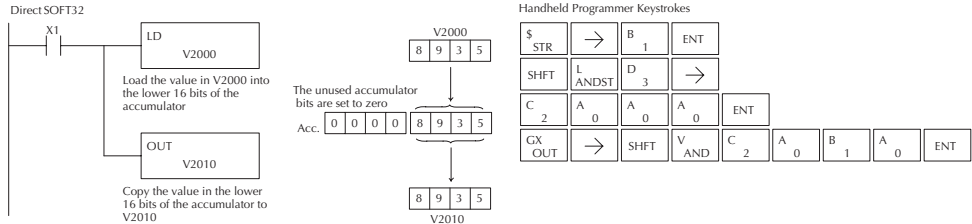
The Out instruction is a 16 bit instruction that copies the value in the lower 16 bits of the accumulator to a specified V memory location (Aaaa).

OUT  
A aaa

Operand Data Type	DL06 Range
..... A	aaa
V memory .....	V See memory map
Pointer .....	P See memory map

Discrete Bit Flags	Description
SP53	On if CPU cannot solve the logic.

In the following example, when X1 is on, the value in V2000 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the lower 16 bits of the accumulator are copied to V2010 using the Out instruction. V2000



Out Double (OUTD)

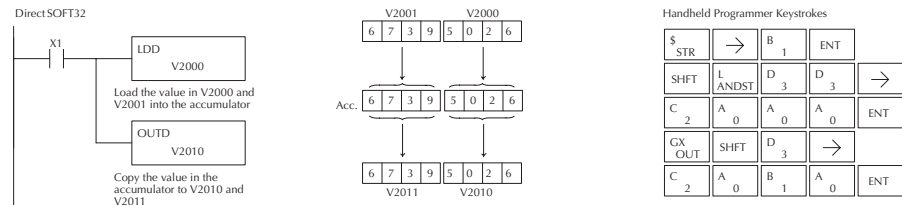
The Out Double instruction is a 32 bit instruction that copies the value in the accumulator to two consecutive V memory locations at a specified starting location (Aaaa).

OUTD  
A aaa

Operand Data Type	DL06 Range
..... A	aaa
V memory .....	V See memory map
Pointer .....	P See memory map

Discrete Bit Flags	Description
SP53	On if CPU cannot solve the logic.

In the following example, when X1 is on, the 32 bit value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.



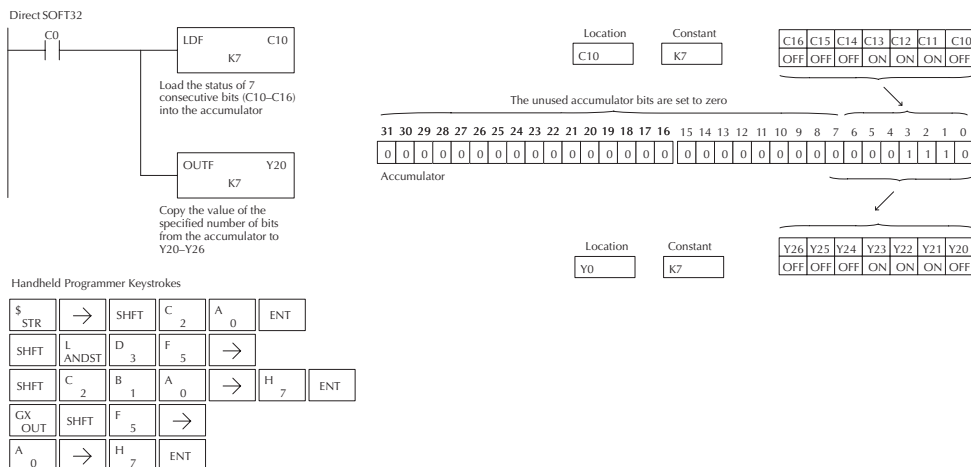
## Out Formatted (OUTF)

The Out Formatted instruction outputs 1–32 bits from the accumulator to the specified discrete memory locations. The instruction requires a starting location (Aaaa) for the destination and the number of bits (Kbbb) to be output.



Operand Data Type	DL06 Range	
..... A	aaa	bbb
Inputs ..... X	0–777	—
Outputs ..... Y	0–777	—
Control Relays ..... C	0–1777	—
Constant ..... K	—	1–32

In the following example, when C0 is on, the binary pattern of C10–C16 (7 bits) will be loaded into the accumulator using the Load Formatted instruction. The lower 7 bits of the accumulator are output to Y0–Y6 using the Out Formatted instruction.



## Pop (POP)

The Pop instruction moves the value from the first level of the accumulator stack (32 bits) to the accumulator and shifts each value in the stack up one level.

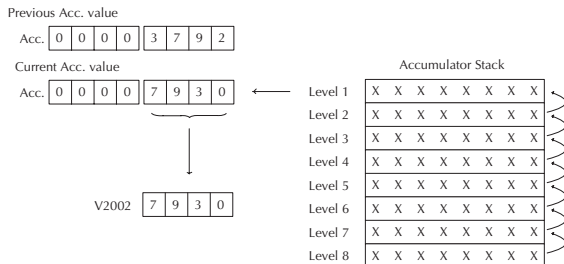
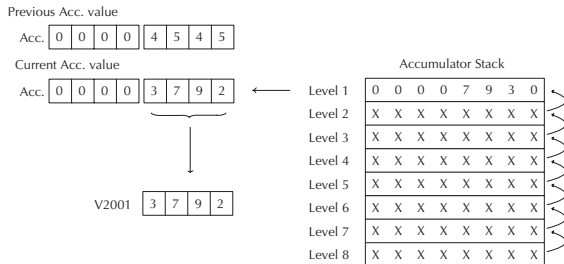
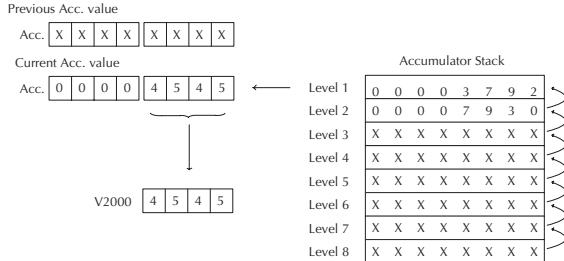
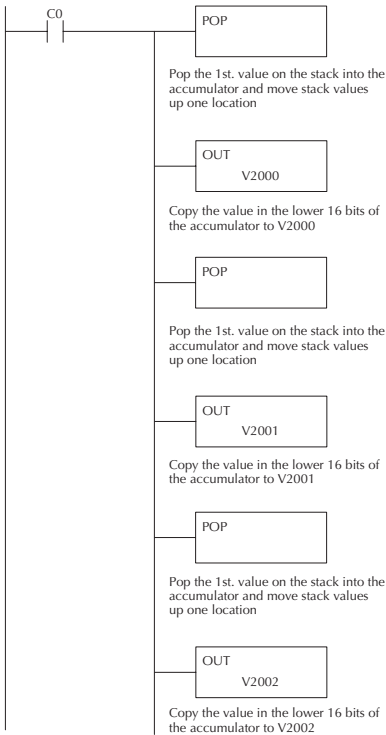


Discrete Bit Flags	Description
SP63	on when the result of the instruction causes the value in the accumulator to be zero.

## Pop Instruction Continued

In the example below, when C0 is on, the value 4545 that was on top of the stack is moved into the accumulator using the Pop instruction. The value is output to V2000 using the Out instruction. The next Pop moves the value 3792 into the accumulator and outputs the value to V2001. The last Pop moves the value 7930 into the accumulator and outputs the value to V2002. Please note if the value in the stack were greater than 16 bits (4 digits) the Out Double instruction would be used and 2 V memory locations for each Out Double must be allocated.

Direct SOFT32



Handheld Programmer Keystrokes

\$ STR	→	SHFT	C 2	A 0	ENT				
SHFT	P CV	SHFT	O INST#	P CV	ENT				
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	A 0	ENT	
SHFT	P CV	SHFT	O INST#	P CV	ENT				
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	B 1	ENT	
SHFT	P CV	SHFT	O INST#	P CV	ENT				
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	C 2	ENT	

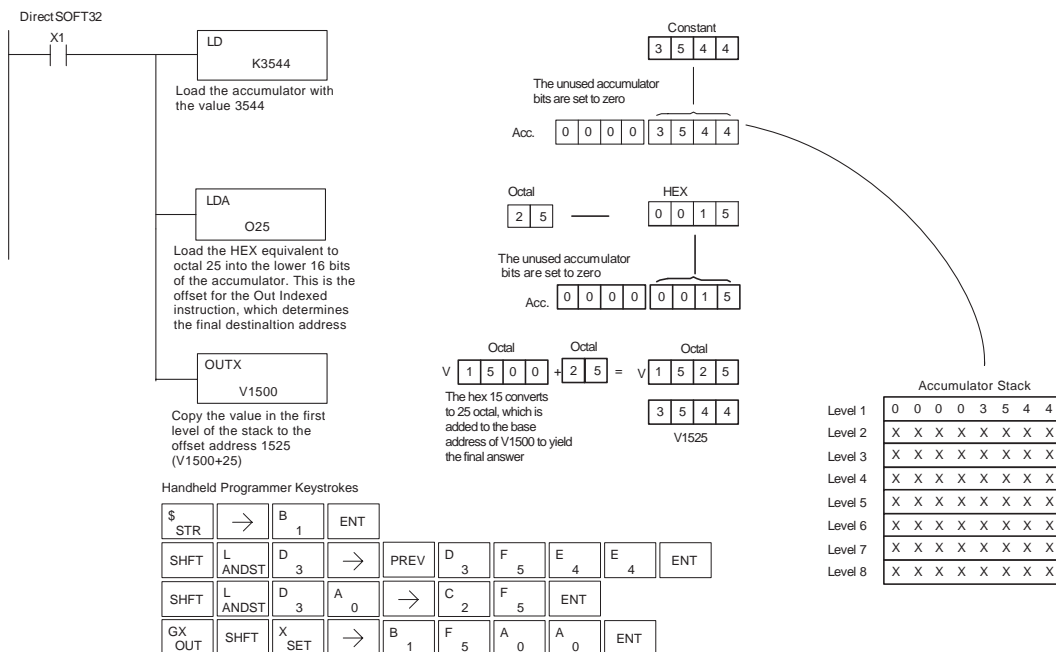
## Out Indexed (OUTX)

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.

OUTX  
A aaa

Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map

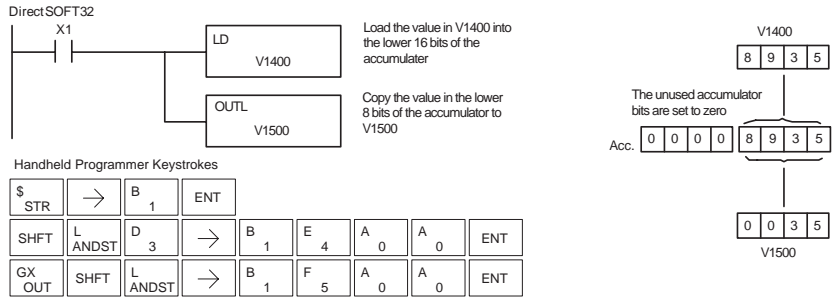
In the following example, when X1 is on, the constant value 3544 is loaded into the accumulator. This is the value that will be output to the specified offset V memory location (V1525). The value 3544 will be placed onto the stack when the Load Address instruction is executed. Remember, two consecutive Load instructions places the value of the first load instruction onto the stack. The Load Address instruction converts octal 25 to HEX 15 and places the value in the accumulator. The Out Indexed instruction outputs the value 3544 which resides in the first level of the accumulator stack to V1525.



Out Least (OUTL)

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.

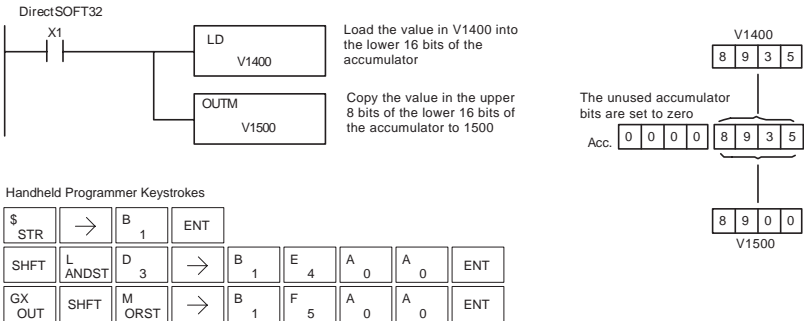


Out Most (OUTM)

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	DL06 Range
..... A	aaa
V memory .....	V
Pointer .....	P

In the following example, when X1 is on, the value in V1400 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the upper 8 bits of the lower 16 bits of the accumulator are copied to V1500 using the Out Most instruction.



# Logical Instructions (Accumulator)

## And (AND)

The And instruction is a 16 bit instruction that logically ands the value in the lower 16 bits of the accumulator with a specified V memory location (Aaaa). The result resides in the accumulator. The discrete status flag indicates if the result of the And is zero.

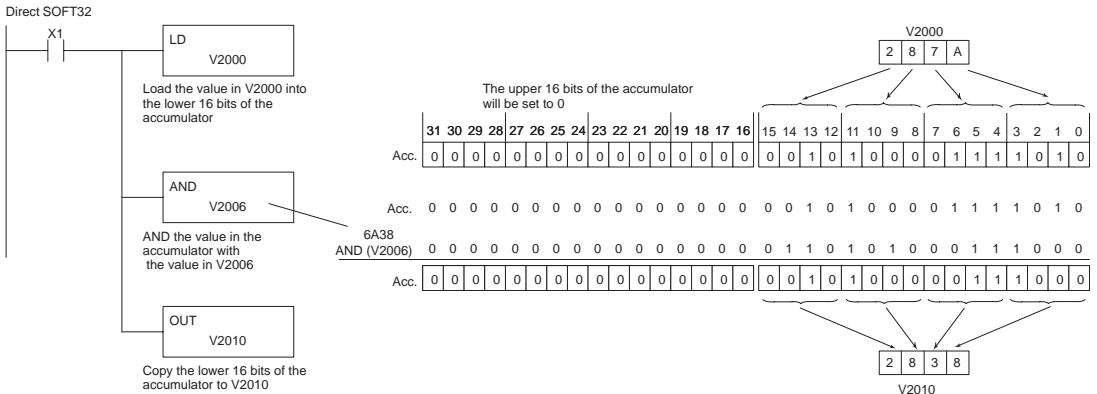


Operand Data Type	DL06 Range
..... A	aaa
V memory .....	See memory map
Pointer .....	See memory map

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero.
SP70	on when the value loaded into the accumulator by any instruction is zero.

**NOTE:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the accumulator is anded with the value in V2006 using the And instruction. The value in the lower 16 bits of the accumulator is output to V2010 using the Out instruction.



### Handheld Programmer Keystrokes

\$	→	B	1	ENT					
SHFT	L	ANDST	D	3	→	C	2	A	0
V	AND	→	SHFT	V	AND	C	2	A	0
GX	OUT	→	SHFT	V	AND	C	2	A	0



And Double (ANDD)

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).



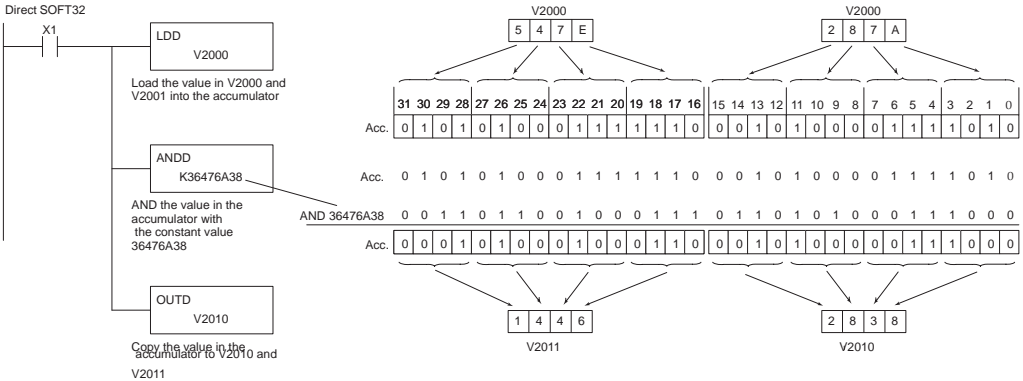
Operand Data Type	DL06 Range
	<b>aaa</b>
V memory .....	See memory map
Pointer .....	See memory map
Constant .....	0-FFFFFFF

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero.
SP70	Will be on if the result in the accumulator is negative



**NOTE:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is anded with 36476A38 using the And Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
SHFT	L ANDST	D 3	D 3
V AND	SHFT	D 3	→
CX OUT	SHFT	D 3	→
		C 2	A 0
		A 0	A 0
		A 0	A 0
		ENT	
		K JMP	D 3
		G 6	E 4
		H 7	G 6
		SHFT	A 0
		SHFT	D 3
		I 8	ENT

## And Formatted (ANDF)

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).

ANDF    Aaaa  
          Kbbb

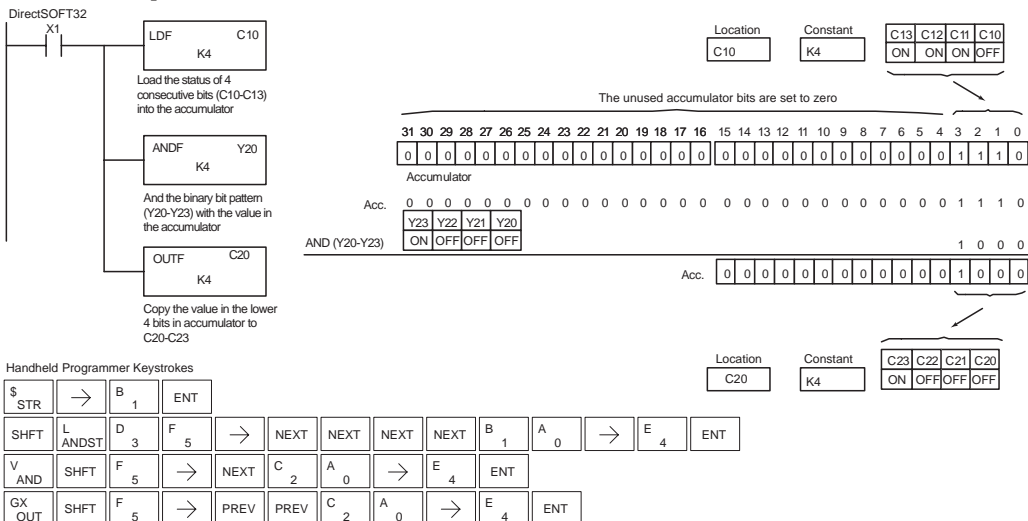
Operand Data Type	DL06 Range	
..... B	<b>aaa</b>	<b>bbb</b>
Inputs ..... X	0-777	-
Outputs ..... Y	0-777	-
Control Relays ..... C	0-1777	-
Stage Bits ..... S	0-1777	-
Timer Bits ..... T	0-377	-
Counter Bits ..... CT	177	-
Special Relays ..... SP	0-777	-
Constant ..... K	-	1-32

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero.
SP70	Will be on if the result in the accumulator is negative



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

In the following example, when X1 is on the Load Formatted instruction loads C10–C13 (4 binary bits) into the accumulator. The accumulator 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.



And with Stack (ANDS)

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).

ANDS

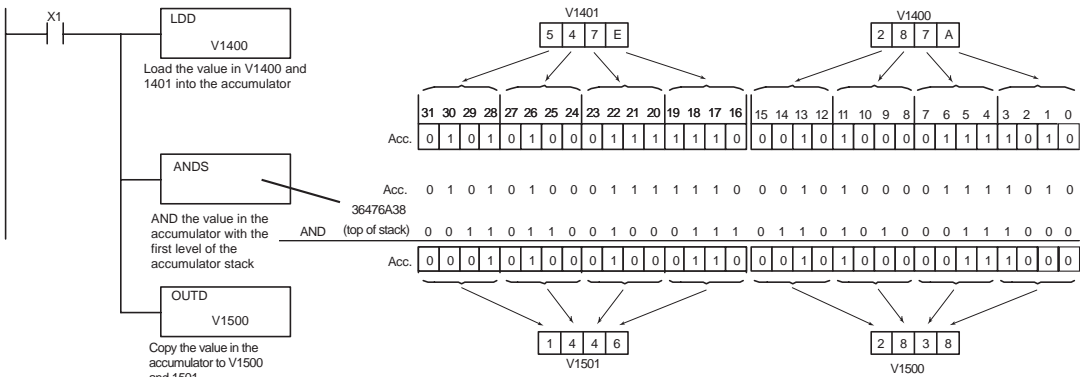
Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero.
SP70	Will be on if the result in the accumulator is negative



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

In the following example when X1 is on, the binary value in the accumulator will be anded with the binary value in the first level of the accumulator stack. The result resides in the accumulator. The 32 bit value is then output to V1500 and V1501.

DirectSOFT32



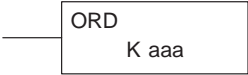
Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
SHIFT	L ANDST	D 3	D 3
V AND	SHIFT	S RST	ENT
GX OUT	SHIFT	D 3	→
		B 1	F 5
		A 0	A 0
			ENT



Or Double (ORD)

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).



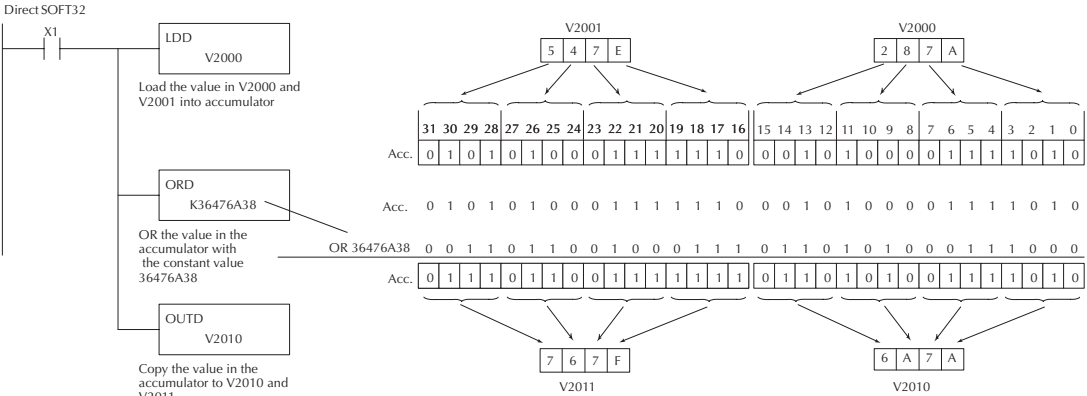
Operand Data Type	DL06 Range
	<b>aaa</b>
V memory .....	V
Pointer .....	P
Constant .....	K
	0-FFFFFFF

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero.
SP70	Will be on if the result in the accumulator is negative

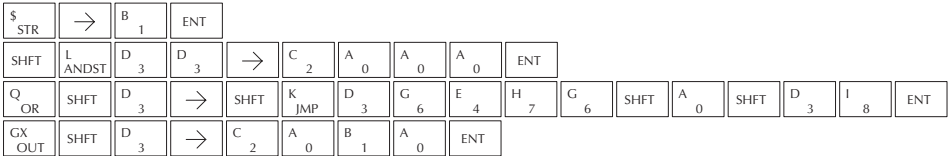


**NOTE:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is ored with 36476A38 using the Or Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes





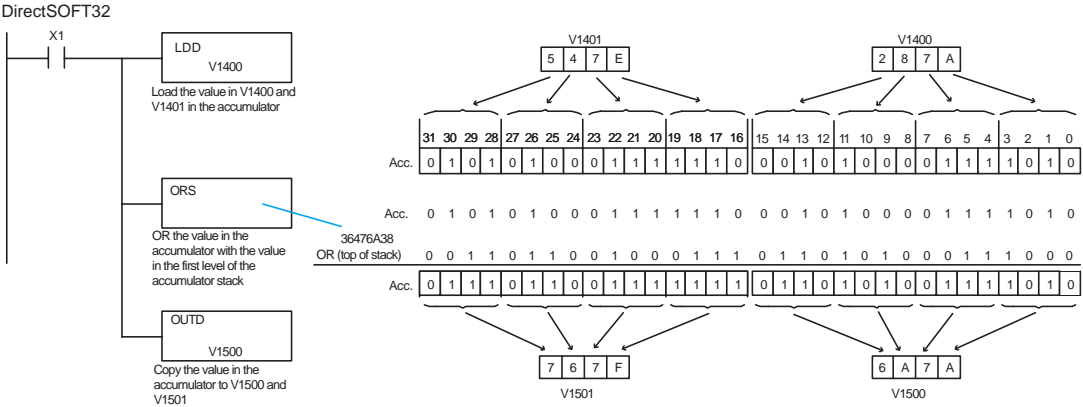
Or with Stack (ORS)

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	on when the value loaded into the accumulator by any instruction is zero.

In the following example when X1 is on, the binary value in the accumulator will be ored with the binary value in the first level of the stack. The result resides in the accumulator.

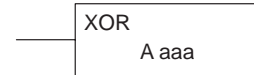


Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT
Q OR	SHFT	S RST	ENT						
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

## Exclusive Or (XOR)

The Exclusive Or instruction is a 16 bit instruction that performs an exclusive or of the value in the lower 16 bits of the accumulator and a specified V memory location (Aaaa). The result resides in the in the accumulator. The discrete status flag indicates if the result of the XOR is zero.

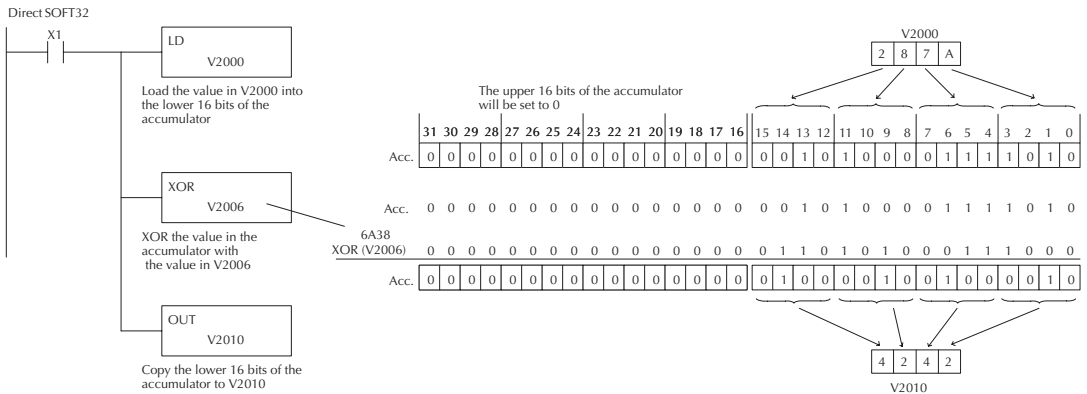


Operand Data Type	DL06 Range
..... A	aaa
V memory .....	See memory map
Pointer .....	See memory map

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero.
SP70	on when the value loaded into the accumulator by any instruction is zero.

**NOTE:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the accumulator is exclusive ored with V2006 using the Exclusive Or instruction. The value in the lower 16 bits of the accumulator are output to V2010 using the Out instruction.



Handheld Programmer Keystrokes

\$ STR	→	SHIFT	X SET	B 1	ENT
SHIFT	L ANDST	D 3	→	SHIFT	V AND
SHIFT	X SET	SHIFT	O OR	→	SHIFT
GX OUT	→	SHIFT	V AND	C 2	A 0
				B 1	A 0
				A 0	ENT
				C 2	A 0
				A 0	ENT
				G 6	ENT



Exclusive Or Double (XORD)

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).

XORD  
K aaa

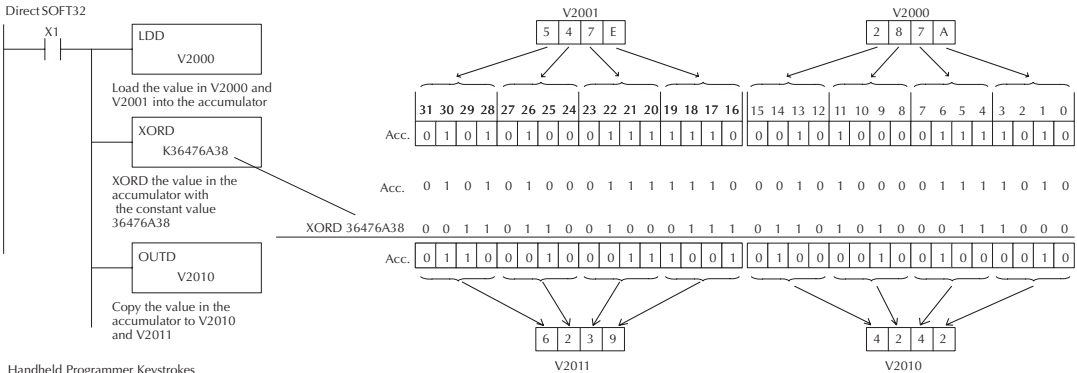
Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0-FFFFFFF

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero.
SP70	Will be on if the result in the accumulator is negative



**NOTE:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is exclusively ored with 36476A38 using the Exclusive Or Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
SHFT	L ANDST	D 3	D 3 → C 2 A 0 A 0 A 0 ENT
SHFT	X SET	Q OR	SHFT D 3 → SHFT K JMP
D 3	G 6	E 4	H 7 G 6 SHFT A 0 SHFT D 3 I 8 ENT
GX OUT	SHFT D 3	→	C 2 A 0 B 1 A 0 ENT

## Exclusive Or Formatted (XORF)

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).

XORF	Aaaa
	K bbb

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

Operand Data Type		DL06 Range	
	A/B	aaa	bbb
Inputs	X	0-777	-
Outputs	Y	0-777	-
Control Relays	C	0-1777	-
Stage Bits	S	0-1777	-
Timer Bits	T	0-377	-
Counter Bits	CT	177	-
Special Relays	SP	0-777	-
Constant	K	-	1-32

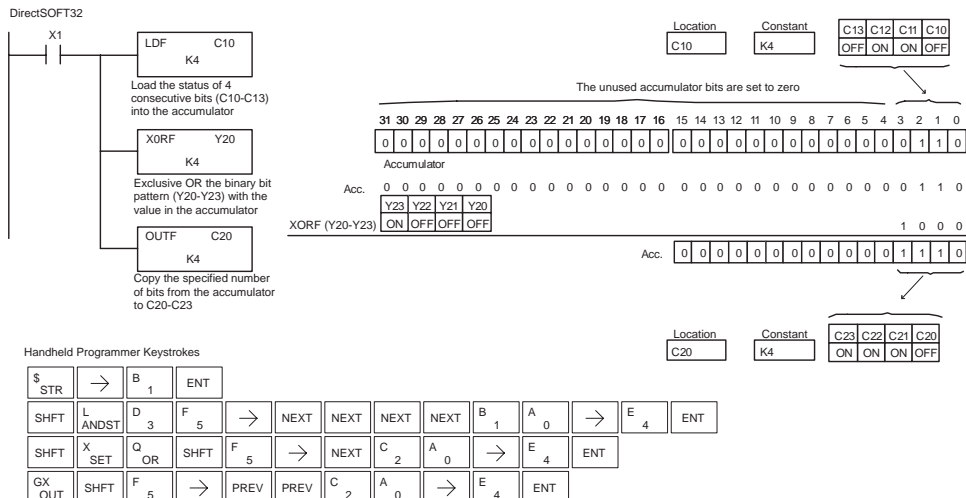
  

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero.
SP70	on when the value loaded into the accumulator by any instruction is zero.



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

In the following example, when X1 is on, the 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 Ored with the bit pattern from Y20–Y23 using the Exclusive Or Formatted instruction. The value in the lower 4 bits of the accumulator are output to C20–C23 using the Out Formatted instruction.



Exclusive Or with Stack (XORS)

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).

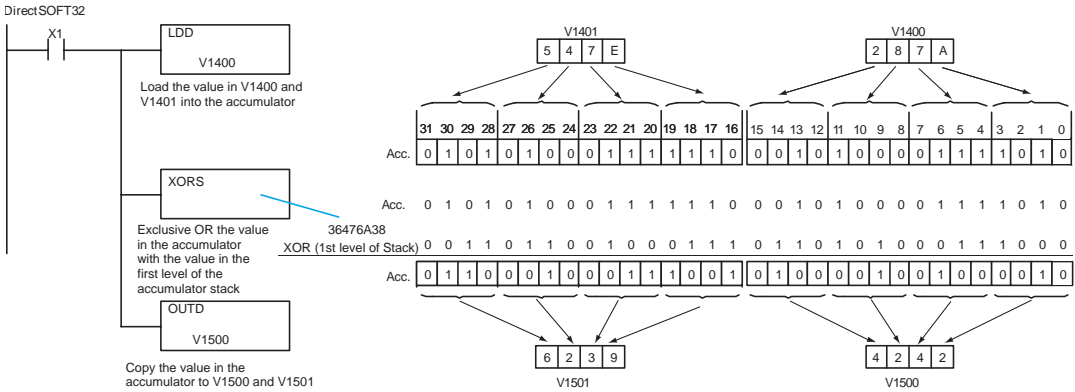
XORS



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

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero.
SP70	on when the value loaded into the accumulator by any instruction is zero.

In the following example when X1 is on, the binary value in the accumulator will be exclusive ored with the binary value in the first level of the accumulator stack. The result will reside in the accumulator.



Handheld Programmer Keystrokes

\$	STR	→	B <sub>1</sub>	ENT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
----	-----	---	----------------	-----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

## Compare (CMP)

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.

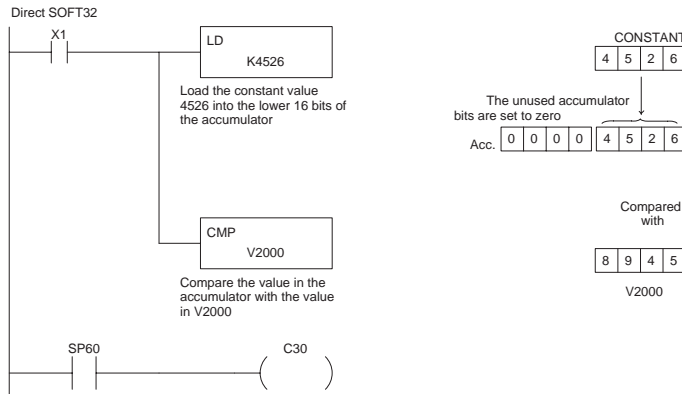
CMP  
A aaa

Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map

Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.

**NOTE:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example when X1 is on, the constant 4526 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the accumulator is compared with the value in V2000 using the Compare instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on energizing C30.

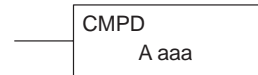


Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT							
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	F 5	C 2	G 6	ENT
SHFT	C 2	SHFT	M ORST	P CV	→	C 2	A 0	A 0	A 0	ENT
\$ STR	→	SHFT	SP STRN	G 6	A 0	ENT				
GX OUT	→	SHFT	C 2	D 3	A 0	ENT				

## Compare Double (CMPD)

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.

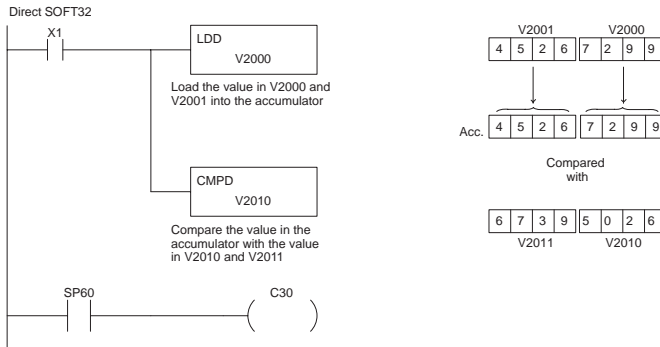


Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0-FFFFFFFF

Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.

**NOTE:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is compared with the value in V2010 and V2011 using the CMPD instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on energizing C30.

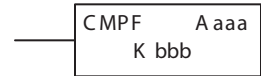


## Handheld Programmer Keystrokes

\$ STR	→	B <sub>1</sub>	ENT									
SHFT	L ANDST	D <sub>3</sub>	D <sub>3</sub>	→	C <sub>2</sub>	A <sub>0</sub>	A <sub>0</sub>	A <sub>0</sub>	ENT			
SHFT	C <sub>2</sub>	SHFT	M ORST	P CV	D <sub>3</sub>	→	C <sub>2</sub>	A <sub>0</sub>	B <sub>1</sub>	A <sub>0</sub>	ENT	
\$ STR	→	SHFT	SP STRN	G <sub>6</sub>	A <sub>0</sub>	ENT						
GX OUT	→	SHFT	C <sub>2</sub>	D <sub>3</sub>	A <sub>0</sub>	ENT						

## Compare Formatted (CMPF)

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.



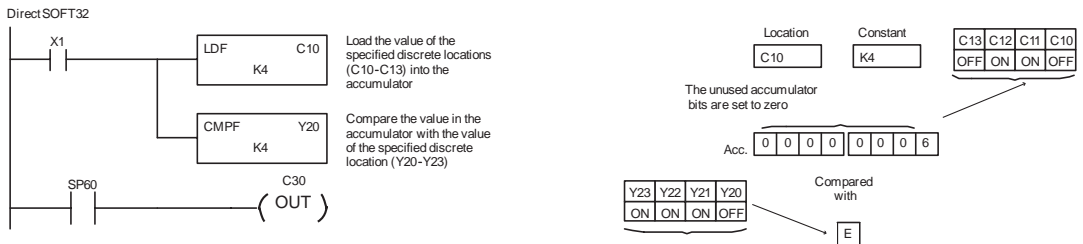
Operand Data Type		DL06 Range	
.....	A/B	aaa	bbb
Inputs .....	X	0-777	-
Outputs .....	Y	0-777	-
Control Relays .....	C	0-1777	-
Stage Bits .....	S	0-1777	-
Timer Bits .....	T	0-377	-
Counter Bits .....	CT	0-177	-
Special Relays .....	SP	0-777	-
Constant .....	K	-	1-32

Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.



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

In the following example, when X1 is on the Load Formatted instruction loads the binary value (6) from C10–C13 into the accumulator. The CMPF instruction compares the value in the accumulator to the value in Y20–Y23 (E hex). The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on energizing C30.



Compare with Stack (CMPS)

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.

CMPS

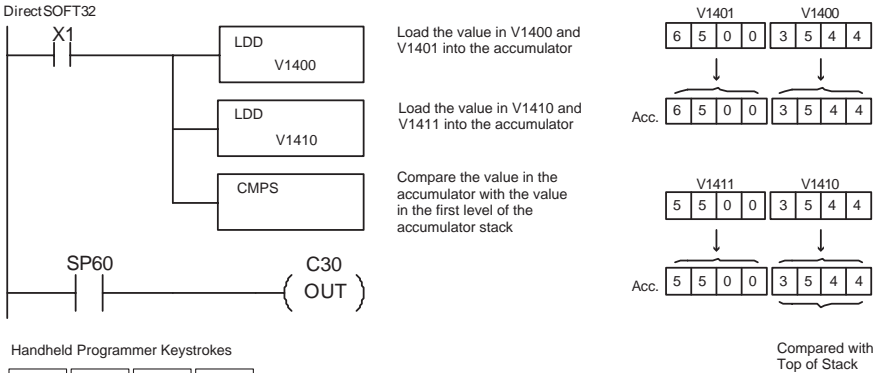
The corresponding status flag will be turned on indicating the result of the comparison. This does not affect the value in the accumulator.

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.



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

In the following example when X1 is on, the value in V1400 and V1401 is loaded into the accumulator using the Load Double instruction. The value in V1410 and V1411 is loaded into the accumulator using the Load Double instruction. The value that was loaded into the accumulator from V1400 and V1401 is placed on top of the stack when the second Load instruction is executed. The value in the accumulator is compared with the value in the first level of the accumulator stack using the CMPS instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value in the stack, SP60 will turn on, energizing C30.

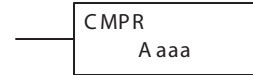


Handheld Programmer Keystrokes

\$	STR	→	B	1	ENT														
SHFT	L	ANDST	D	3	D	3	→	B	1	E	4	A	0	A	0	ENT			
SHFT	L	ANDST	D	3	D	3	→	B	1	E	4	B	1	A	0	ENT			
SHFT	C	2	SHFT	M	ORST	P	CV	S	RST	ENT									
\$	STR	PREV	G	6	A	0	ENT												
GX	OUT	→	NEXT	NEXT	NEXT	SHFT	C	2	D	3	A	0	ENT						

## Compare Real Number (CMPR)

The Compare Real Number instruction compares a real number value in the accumulator with two consecutive V memory locations containing a real number. The corresponding status flag will be turned on indicating the result of the comparison. Both numbers being compared are 32 bits long.



Operand Data Type	DL06 Range
..... A	<b>aaa</b>
V memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... R	-3.402823E+038 to + 3.402823E+038

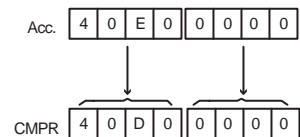
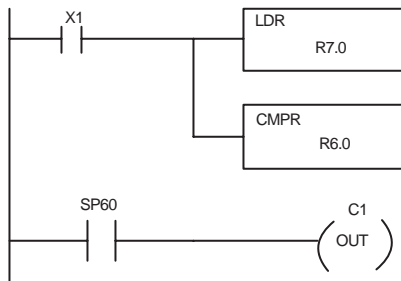
Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.
SP71	On anytime the V-memory specified by a pointer (P) is not valid
SP75	On if a BCD number is expected and a non-BCD number is encountered.



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

In the following example when X1 is on, the 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 SP60).

DirectSOFT32





# Math Instructions

## Add (ADD)

Add is a 16 bit instruction that adds a BCD value in the accumulator with a BCD value in a V memory location (Aaaa). The result resides in the accumulator.

ADD  
A aaa

Operand Data Type	DL06 Range
..... A	aaa
V memory .....	See memory map
Pointer .....	See memory map

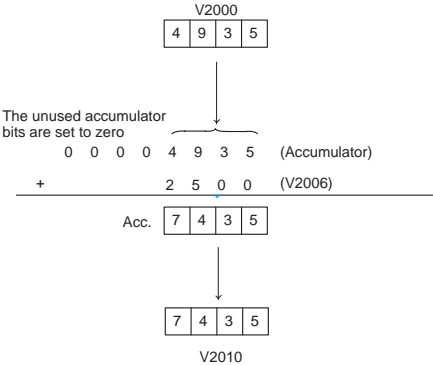
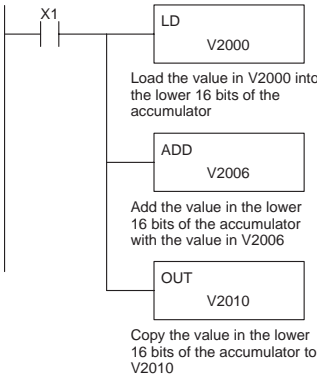
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16 bit addition instruction results in a carry.
SP67	On when the 32 bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.



**NOTE:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the lower 16 bits of the accumulator are added to the value in V2006 using the Add instruction. The value in the accumulator is copied to V2010 using the Out instruction.

Direct SOFT32

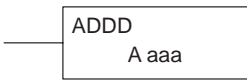


Handheld Programmer Keystrokes

\$	STR	→	B	1	ENT										
SHFT	L	ANDST	D	3	→	C	2	A	0	A	0	A	0	ENT	
SHFT	A	0	D	3	D	3	→	C	2	A	0	A	0	G	6
GX	OUT	→	SHFT	V	AND	C	2	A	0	B	1	A	0	ENT	

### Add Double (ADDD)

Add Double is a 32 bit instruction that adds the BCD value in the accumulator with a BCD value (Aaaa), which is either two consecutive V memory locations or an 8-digit (max.) BCD constant. The result resides in the accumulator.



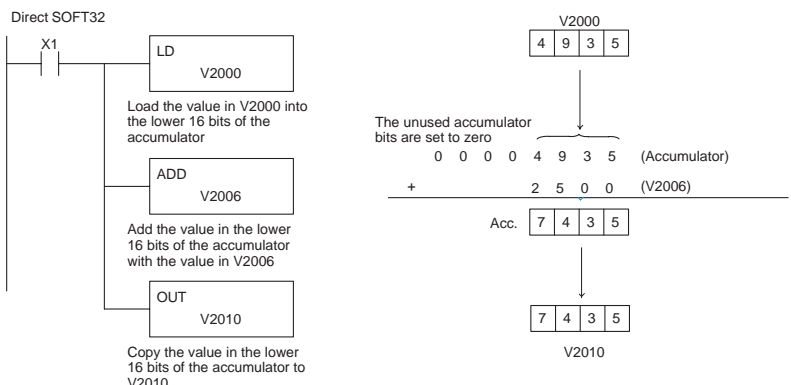
Operand Data Type	DL06Range
..... A	<b>aaa</b>
V memory .....	See memory map
Pointer .....	See memory map
Constant .....	0-99999999

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:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is added with the value in V2006 and V2007 using the Add Double instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



#### Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
--------	---	-----	-----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Add Real (ADDR)

The Add Real instruction adds a real number in the accumulator with either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must conform to the IEEE floating point format.

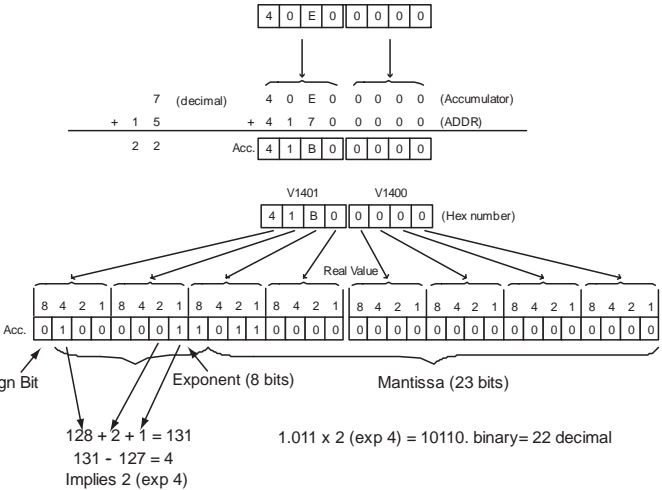
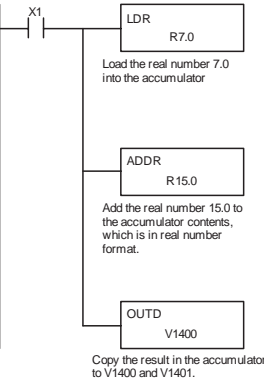
ADDR  
A aaa

Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map
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 an invalid floating point number.
SP73	on when a signed addition or subtraction results in a incorrect sign bit.
SP74	On anytime a floating point math operation results in an underflow error.
SP75	On if a BCD number is expected and a non-BCD number is encountered.

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

DirectSOFT32



NOTE: The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use DirectSOFT32 for this feature.

## Subtract (SUB)

Subtract is a 16 bit instruction that subtracts the BCD value (Aaaa) in a V memory location from the BCD value in the lower 16 bits of the accumulator. The result resides in the accumulator.

SUB  
A aaa

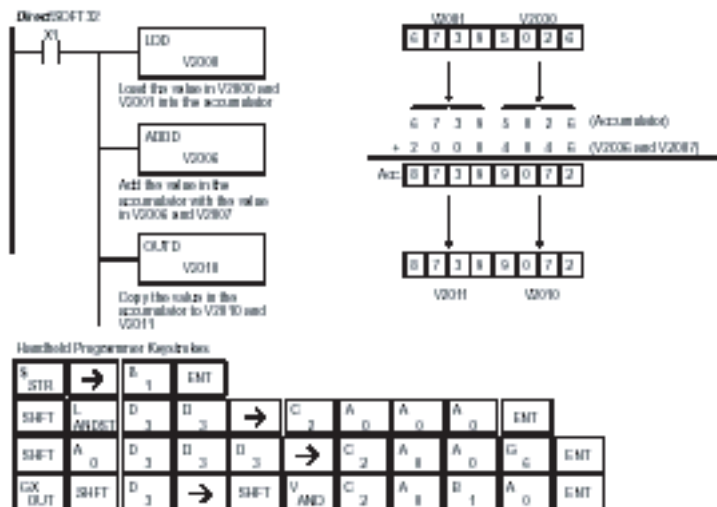
Operand Data Type	DL06Range
..... A	<b>aaa</b>
V memory .....	See memory map
Pointer .....	See memory map

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16 bit subtraction instruction results in a borrow
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:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in V2006 is subtracted from the value in the accumulator using the Subtract instruction. The value in the accumulator is copied to V2010 using the Out instruction.



### Subtract Double (SUBD)

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.

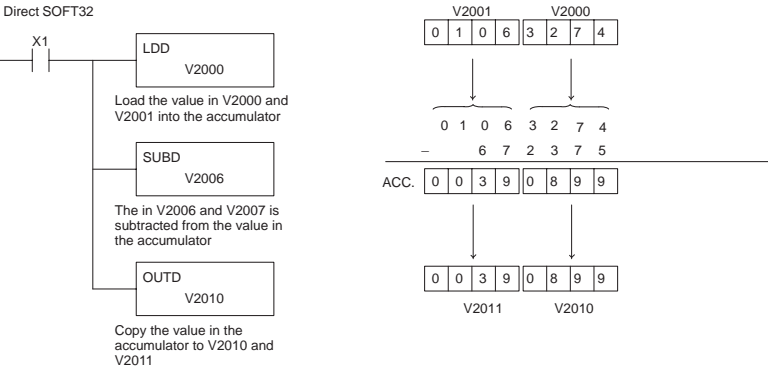
SUBD  
A aaa

Operand Data Type	DL06 Range
..... A	<b>aaa</b>
V memory .....	See memory map
Pointer .....	See memory map
Constant .....	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:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in V2006 and V2007 is subtracted from the value in the accumulator. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT																
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT										
SHFT	S RST	SHFT	U ISG	B 1	D 3	→	C 2	A 0	A 0	G 6	ENT								
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT											

## Subtract Real (SUBR)

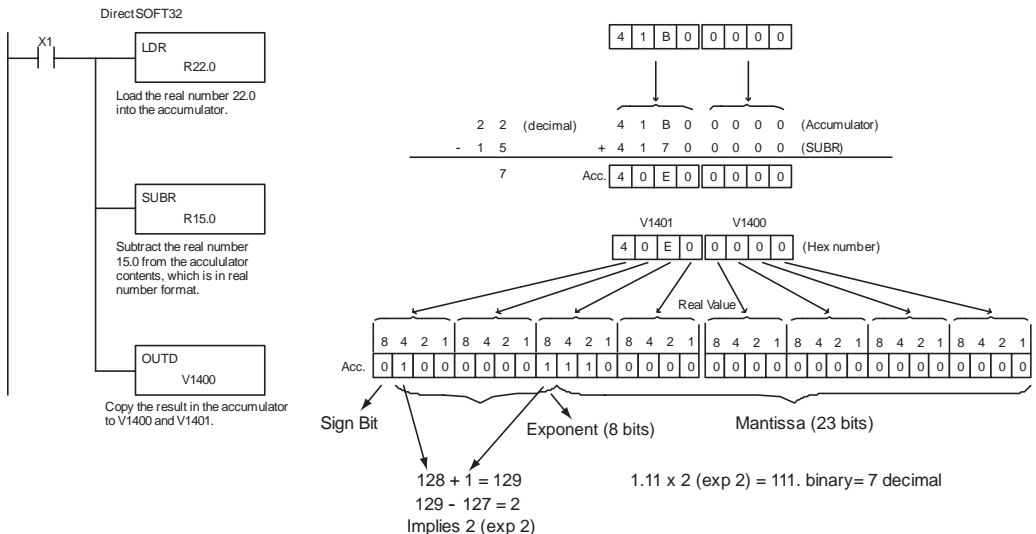
The Subtract Real instruction subtracts a real number in the accumulator from either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must conform to the IEEE floating point format.

SUBR  
A aaa

Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map
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 an invalid floating point number.
SP73	On when a signed addition or subtraction results in a incorrect sign bit.
SP74	On anytime a floating point math operation results in an underflow error.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

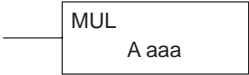
**NOTE:** Status flags are valid only until another instruction uses the same flag.



**NOTE:** The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use DirectSOFT32 for this feature

Multiply (MUL)

Multiply is a 16 bit instruction that multiplies the BCD value (Aaaa), which is either a V memory location or a 4-digit (max.) constant, by the BCD value in the lower 16 bits of the accumulator. The result can be up to 8 digits and resides in the accumulator.



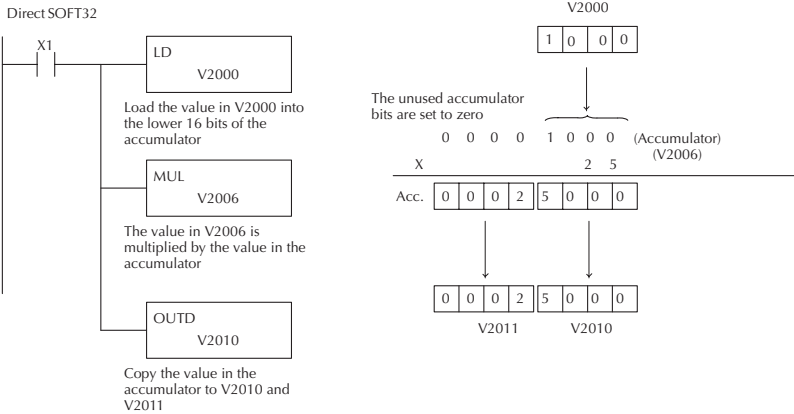
Operand Data Type	DL06 Range
..... A	<b>aaa</b>
V memory .....	See memory map
Pointer .....	See memory map
Constant .....	0-9999

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:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in V2006 is multiplied by the value in the accumulator. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT	
SHFT	M ORST	U ISG	L ANDST	→	C 2	A 0	A 0	G 6	ENT
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

## Multiply Double (MULD)

Multiply Double is a 32 bit instruction that multiplies the 8-digit BCD value in the accumulator by the 8-digit BCD value in the two consecutive V-memory locations specified in the instruction. The lower 8 digits of the results reside in the accumulator. Upper digits of the result reside in the accumulator stack.

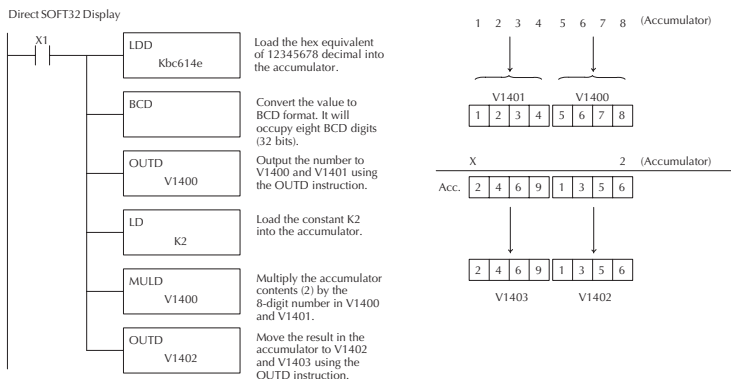
MULD  
A aaa

Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

In the following example, when X1 is on, the constant Kbc614e hex will be loaded into the accumulator. When converted to BCD the number is "12345678". That numbers stored in V1400 and V1401. After loading the constant K2 into the accumulator, we multiply it times 12345678, which is 24691356.



Handheld Programmer Keystrokes

\$	→	B <sub>1</sub>	ENT																						
SHFT	L ANDST	D <sub>3</sub>	D <sub>3</sub>	→	PREV	SHFT	B <sub>1</sub>	C <sub>2</sub>	SHFT	G <sub>6</sub>	B <sub>1</sub>	E <sub>4</sub>	SHFT	E <sub>4</sub>	ENT										
SHFT	B <sub>1</sub>	C <sub>2</sub>	D <sub>3</sub>	ENT																					
GX OUT	SHFT	D <sub>3</sub>	→	B <sub>1</sub>	E <sub>4</sub>	A <sub>0</sub>	A <sub>0</sub>	ENT																	
SHFT	L ANDST	D <sub>3</sub>	→	PREV	C <sub>2</sub>	ENT																			
SHFT	M ORST	U ISG	L ANDST	D <sub>3</sub>	→	B <sub>1</sub>	E <sub>4</sub>	A <sub>0</sub>	A <sub>0</sub>	ENT															
GX OUT	SHFT	D <sub>3</sub>	→	B <sub>1</sub>	E <sub>4</sub>	A <sub>0</sub>	C <sub>2</sub>	ENT																	



Multiply Real (MULR)

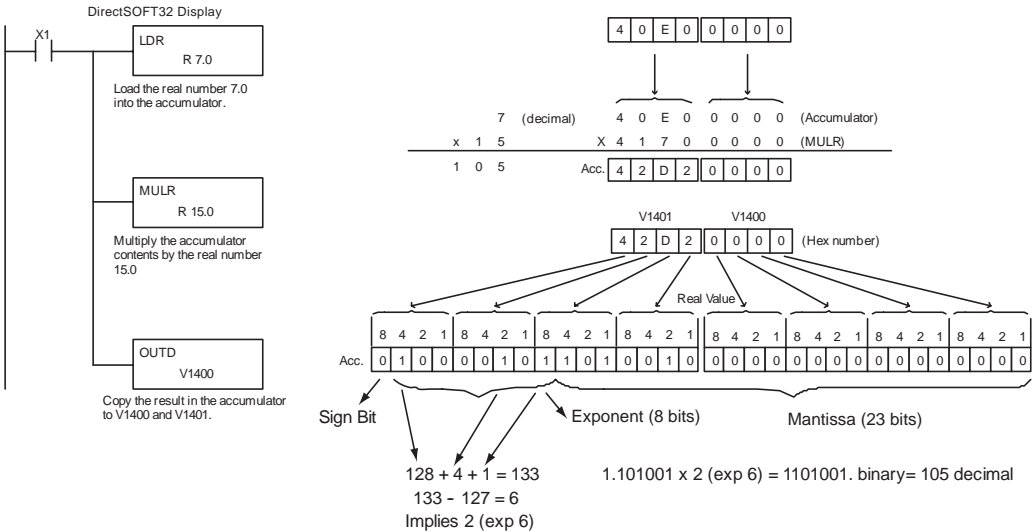
The Multiply Real instruction multiplies a real number in the accumulator with either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must conform to the IEEE floating point format.

MULR  
A aaa

Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map
Real 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 an invalid floating point number.
SP73	On when a signed addition or subtraction results in a incorrect sign bit.
SP74	On anytime a floating point math operation results in an underflow error.
SP75	On when a real number instruction is executed and a non-real number was encountered.

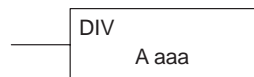
**NOTE:** Status flags are valid only until another instruction uses the same flag.



**NOTE:** The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use DirectSOFT32 for this feature.

# Divide (DIV)

Divide is a 16 bit instruction that divides the BCD value in the accumulator by a BCD value (Aaaa), which is either a V memory location or a 4-digit (max.) constant. The first part of the quotient resides in the accumulator and the remainder resides in the first stack location.



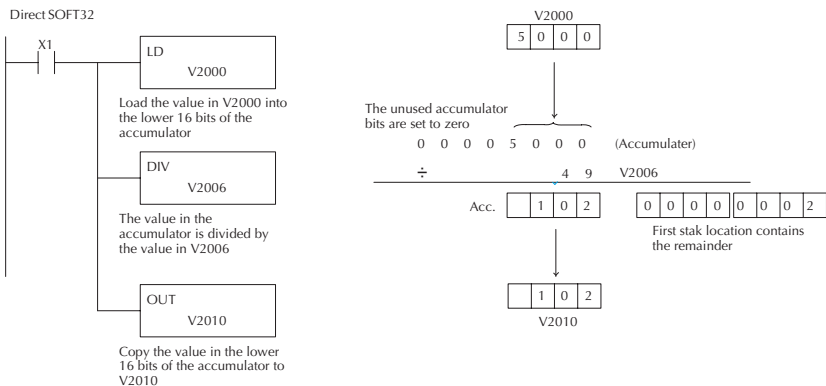
Operand Data Type	DL06 Range
..... A	<b>aaa</b>
V memory .....	See memory map
Pointer .....	See memory map
Constant .....	0-9999

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:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the accumulator will be divided by the value in V2006 using the Divide instruction. The value in the accumulator is copied to V2010 using the Out instruction.



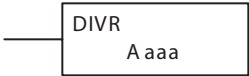
Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT	
SHFT	D 3	I 8	V AND	→	C 2	A 0	A 0	G 6	ENT
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT	



Divide Real (DIVR)

The Divide Real instruction divides a real number in the accumulator by either a real constant or a real number occupying two consecutive V-memory locations. The result resides in the accumulator. Both numbers must conform to the IEEE floating point format.

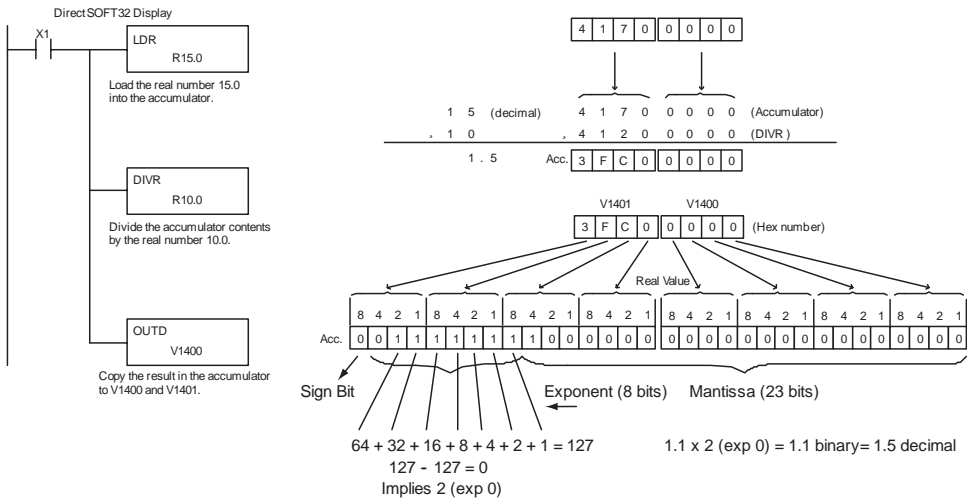


Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map
Real Constant ..... R	-3.402823E+038 to + -3.402823E+038

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 an invalid floating point number.
SP73	On when a signed addition or subtraction results in a incorrect sign bit.
SP74	On anytime a floating point math operation results in an underflow error.
SP75	On when a real number instruction is executed and a non-real number was encountered.



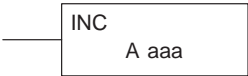
**NOTE:** Status flags are valid only until another instruction uses the same flag.



**NOTE:** The current HPP does not support real number entry with automatic conversion to the 32-bit IEEE format. You must use DirectSOFT32 for this feature.

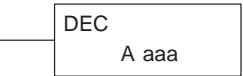
Increment (INC)

The Increment instruction increments a BCD value in a specified V memory location by “1” each time the instruction is executed.



Decrement (DEC)

The Decrement instruction decrements a BCD value in a specified V memory location by “1” each time the instruction is executed.



Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map

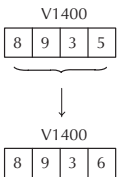
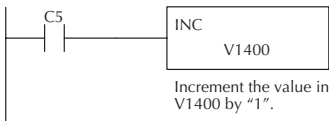
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.



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

In the following increment example, when C5 is on the value in V1400 increases by one.

Direct SOFT32

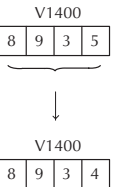
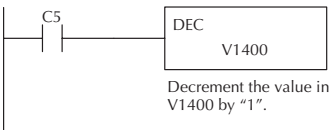


Handheld Programmer Keystrokes

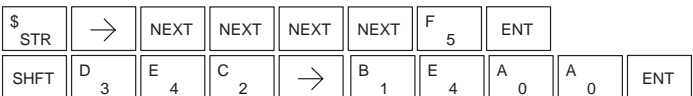


In the following decrement example, when C5 is on the value in V1400 is decreased by one.

Direct SOFT32



Handheld Programmer Keystrokes



Add Binary (ADDB)

Add Binary is a 16 bit instruction that adds the unsigned 2's complement binary value in the lower 16 bits of the accumulator with an unsigned 2's complement binary value (Aaaa), which is either a V memory location or a 16-bit constant. The result can be up to 32 bits (unsigned 2's complement) and resides in the accumulator.



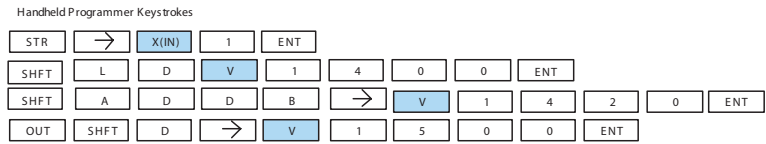
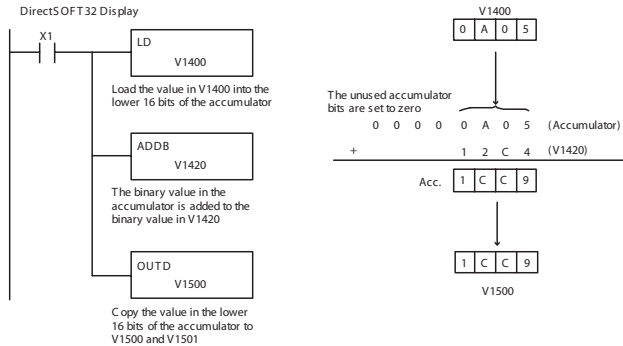
Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0-FFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16-bit addition instruction results in a carry.
SP67	On when the 32-bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP73	On when a signed addition or subtraction results in an incorrect sign bit.



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

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The binary value in the accumulator will be added to the binary value in V1420 using the Add Binary instruction. The value in the accumulator is copied to V1500 and V1501 using the Out instruction.



### Add Binary Double (ADDBD)

Add Binary Double is a 32 bit instruction that adds the unsigned 2's complement binary value in the accumulator with the value (Aaaa), which is either two consecutive V memory locations or 32-bit unsigned 2's complement binary constant. The result resides in the accumulator.

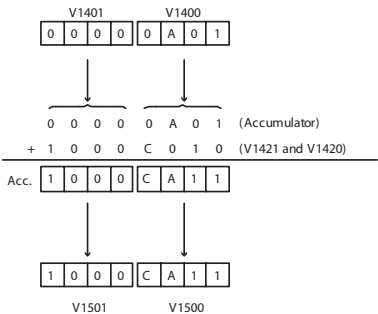
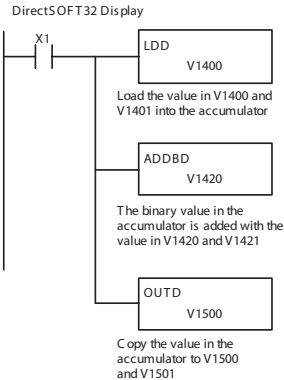
ADDBD  
A aaa

Operand Data Type	DL06 Range
..... A	aaa
V memory .....	See memory map
Pointer .....	See memory map
Constant .....	0-FFFF FFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16-bit addition instruction results in a carry.
SP67	On when the 32-bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP73	On when a signed addition or subtraction results in an incorrect sign bit.

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

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The binary value in the accumulator is added with the binary value in V1420 and V1421 using the Add Binary Double instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT									
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT			
SHFT	A 0	D 3	D 3	B 1	D 3	→	B 1	E 4	C 2	A 0	ENT	
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT				

Subtract Binary (SUBB)

Subtract Binary is a 16 bit instruction that subtracts the unsigned 2’s complement binary value (Aaaa), which is either a V memory location or a 16-bit 2’s complement binary value, from the binary value in the accumulator. The result resides in the accumulator.

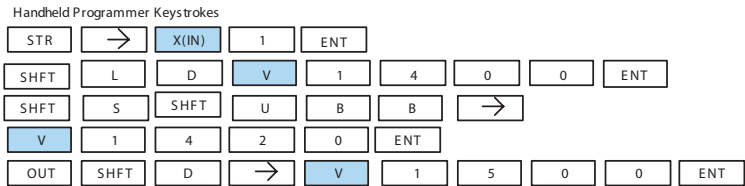
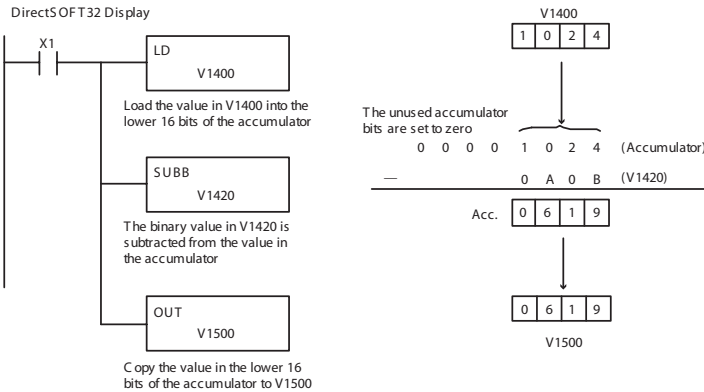
SUBB  
Aaaa

Operand Data Type	DL06 Range
..... A	<b>aaa</b>
V memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0-FFFF

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16-bit subtraction instruction results in a borrow.
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)

Subtract Binary Double is a 32 bit instruction that subtracts the unsigned 2's complement binary value (Aaaa), which is either two consecutive V memory locations or a 32-bit unsigned 2's complement binary constant, from the binary value in the accumulator. The result resides in the accumulator.

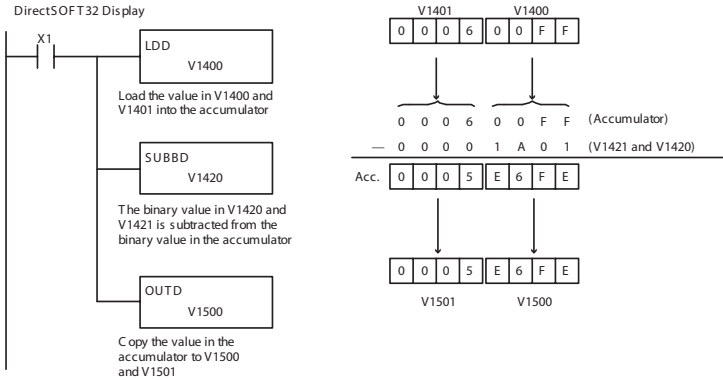
SUBBD  
Aaaa

Operand Data Type	DL06 Range
.....A	<b>aaa</b>
V memory .....	See memory map
Pointer .....	See memory map
Constant .....	0-FFFF 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 and V1401 will be loaded into the accumulator using the Load Double instruction. The binary value in V1420 and V1421 is subtracted from the binary value in the accumulator using the Subtract Binary Double instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

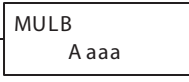


Handheld Programmer Keystrokes

\$	→	B <sub>1</sub>	ENT																	
SHFT	L ANDST	D <sub>3</sub>	D <sub>3</sub>	→	B <sub>1</sub>	E <sub>4</sub>	A <sub>0</sub>	A <sub>0</sub>	ENT											
SHFT	S RST	SHFT	U ISG	B <sub>1</sub>	B <sub>1</sub>	D <sub>3</sub>	→	B <sub>1</sub>	E <sub>4</sub>	C <sub>2</sub>	A <sub>0</sub>	ENT								
GX OUT	SHFT	D <sub>3</sub>	→	B <sub>1</sub>	F <sub>5</sub>	A <sub>0</sub>	A <sub>0</sub>	ENT												

# Multiply Binary (MULB)

Multiply Binary is a 16 bit instruction that multiplies the unsigned 2's complement binary value (Aaaa), which is either a V memory location or a 16-bit unsigned 2's complement binary constant, by the 16-bit binary value in the accumulator. The result can be up to 32 bits and resides in the accumulator.



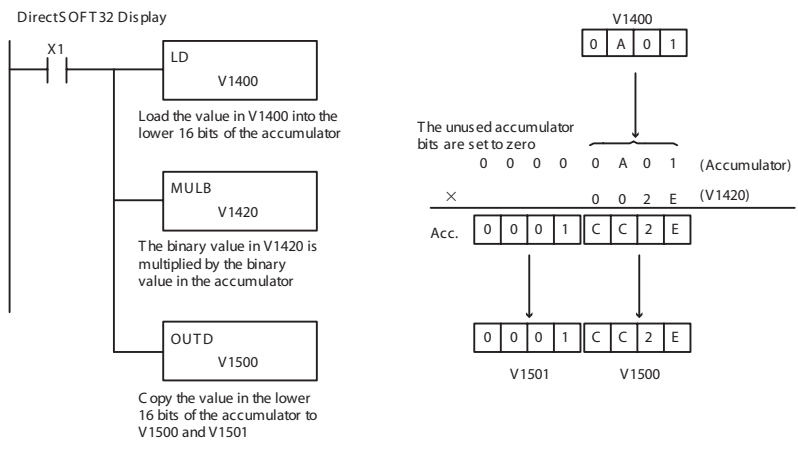
Operand Data Type	DL06 Range
.....A	aaa
V memory .....	See memory map
Pointer .....	See memory map
Constant .....	0-FFFF

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.

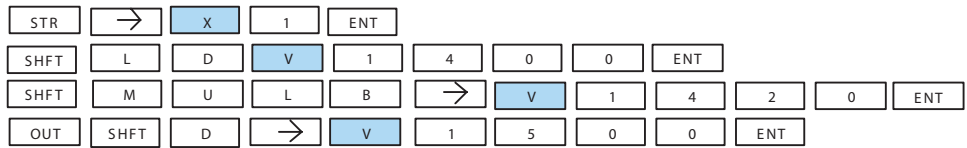


**NOTE:** Status flags are valid only until another instruction uses the same flag.

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.



Handheld Programmer Keystrokes



Divide Binary (DIVB)

Divide Binary is a 16 bit instruction that divides the unsigned 2's complement binary value in the accumulator by a binary value (Aaaa), which is either a V memory location or a 16-bit unsigned 2's complement binary constant. The first part of the quotient resides in the accumulator and the remainder resides in the first stack location.

DIVB  
A aaa

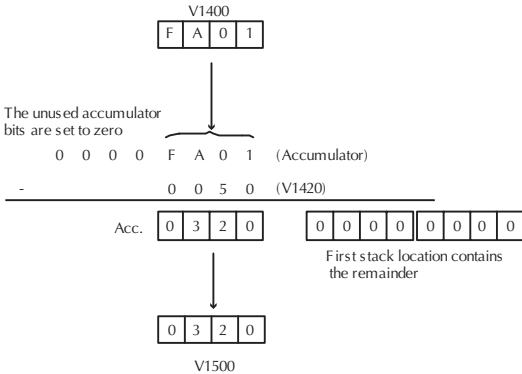
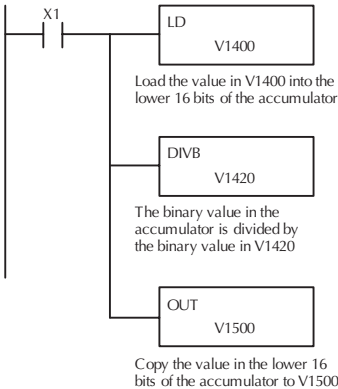
Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0-FFFF

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.

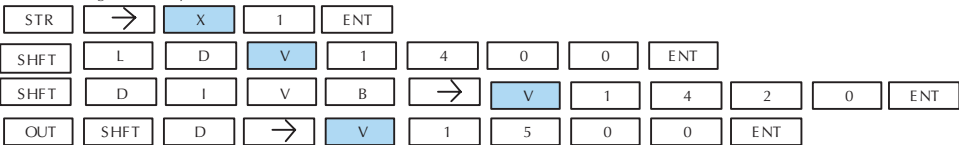
**NOTE:** Status flags are valid only until another instruction uses the same flag.

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The binary value in the accumulator is divided by the binary value in V1420 using the Divide Binary instruction. The value in the accumulator is copied to V1500 using the Out instruction.

DirectSOFT32 Display

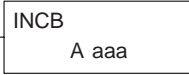


Handheld Programmer Keystrokes



Increment Binary (INCB)

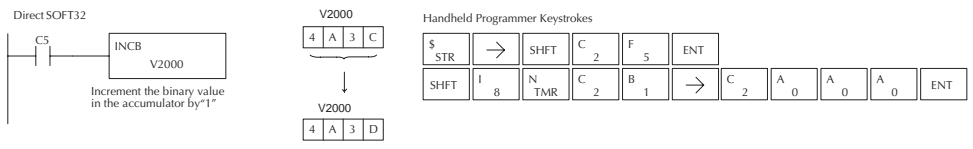
The Increment Binary instruction increments a binary value in a specified V memory location by “1” each time the instruction is executed.



Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map

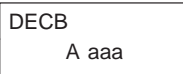
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.

In the following example when C5 is on, the binary value in V2000 is increased by 1.



Decrement Binary (DECB)

The Decrement Binary instruction decrements a binary value in a specified V memory location by “1” each time the instruction is executed.



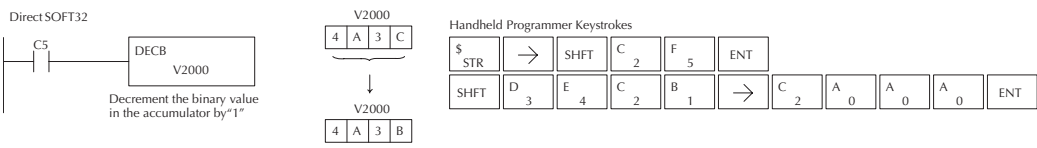
Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map

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



**NOTE:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example when C5 is on, the value in V2000 is decreased by 1.



### Add Formatted (ADDF)

Add Formatted is a 32 bit instruction that adds the BCD value in the accumulator with the BCD value (Aaaa) which is a range of discrete bits. The specified range (Kbbb) can be 1 to 32 consecutive bits. The result resides in the accumulator.

ADDF     A aaa  
          K bbb

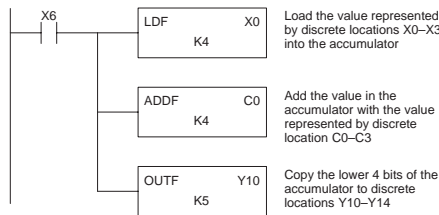
Operand Data Type		DL06 Range	
	A	aaa	bbb
Inputs	X	0-777	—
Outputs	Y	0-777	—
Control Relays	C	0-1777	—
Stage Bits	S	0-1777	—
Timer Bits	T	0-377	—
Counter Bits	CT	0-177	—
Special Relays	SP	0-137 320-717	—
Global I/O	GX	0-3777	—
Constant	K	—	1-32

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16 bit addition instruction results in a carry.
SP67	On when the 32 bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

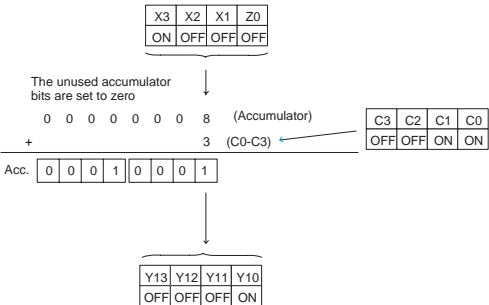
**NOTE:** Status flags are valid only until another instruction uses the same flag.

In the following example, when X6 is on, the value formed by discrete locations X0–X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete locations C0–C3 is added to the value in the accumulator using the Add Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10–Y13 using the Out Formatted instruction.

Direct SOFT32 Display



Handheld Programmer Keystrokes



## Subtract Formatted (SUBF)

Subtract Formatted is a 32 bit instruction that subtracts the BCD value (Aaaa), which is a range of discrete bits, from the BCD value in the accumulator. The specified range (Kbbb) can be 1 to 32 consecutive bits. The result resides in the accumulator.

SUBF      A aaa  
            K bbb

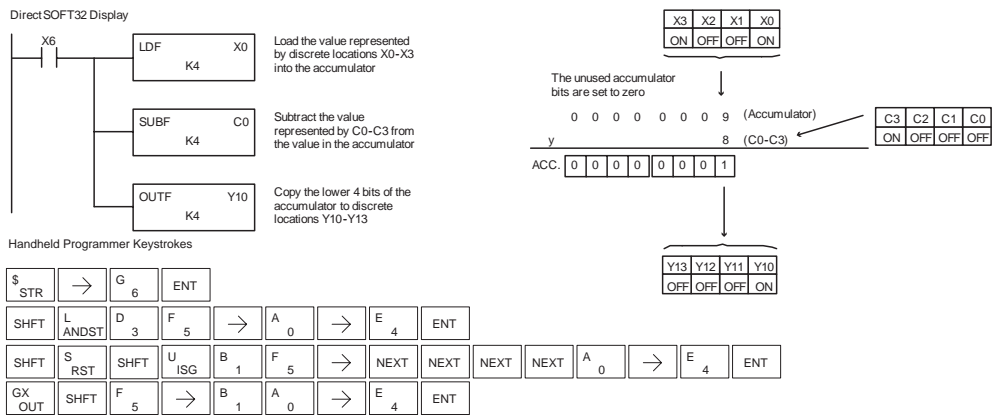
Operand Data Type		DL06 Range	
	A	aaa	bbb
Inputs .....	X	0-777	—
Outputs .....	Y	0-777	—
Control Relays .....	C	0-1777	—
Stage Bits .....	S	0-1777	—
Timer Bits .....	T	0-377	—
Counter Bits .....	CT	0-177	—
Special Relays .....	SP	0-137 320-717	—
Global I/O .....	GX	0-3777	—
Constant .....	K	—	1-32

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
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 any time the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.



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

In the following example, when X6 is on, the value formed by discrete locations X0–X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete 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



Multiply Formatted (MULF)

Multiply Formatted is a 16 bit instruction that multiplies the BCD value in the accumulator by the BCD value (Aaaa) which is a range of discrete bits. The specified range (Kbbb) can be 1 to 16 consecutive bits. The result resides in the accumulator.

MULF     A aaa  
           K bbb

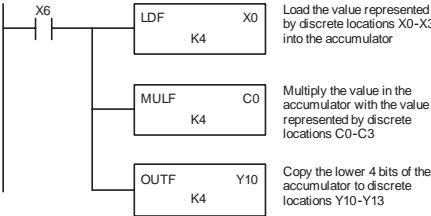
Operand Data Type		DL06 Range	
	A	aaa	bbb
Inputs .....	X	0-777	—
Outputs .....	Y	0-777	—
Control Relays .....	C	0-1777	—
Stage Bits .....	S	0-1777	—
Timer Bits .....	T	0-377	—
Counter Bits .....	CT	0-177	—
Special Relays .....	SP	0-137 320-717	—
Global I/O .....	GX	0-3777	—
Constant .....	K	—	1-16

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On any time the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

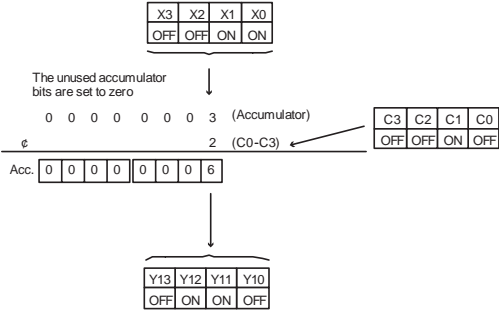
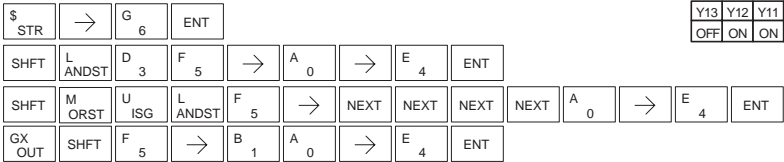
**NOTE:** Status flags are valid only until another instruction uses the same flag.

In the following example, when X6 is on, the value formed by discrete locations X0-X3 is loaded into the accumulator using the Load Formatted instruction. The value formed by discrete locations C0-C3 is multiplied by the value in the accumulator using the Multiply Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10-Y13 using the Out Formatted instruction.

Direct SOFT32 Display



Handheld Programmer Keystrokes



Divide Formatted (DIVE)

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.

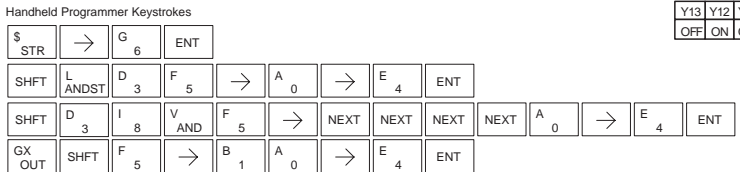
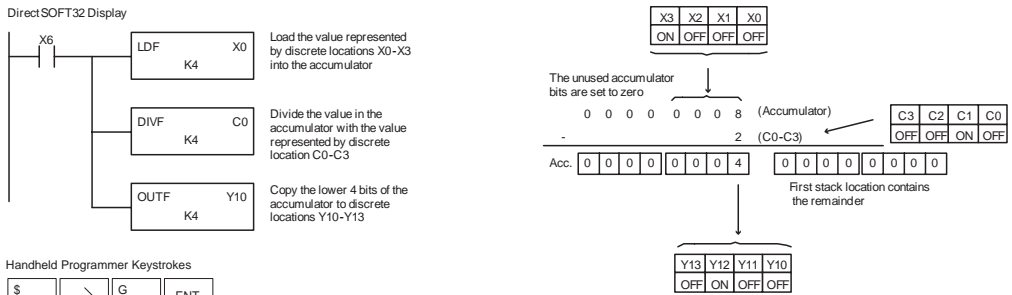


Operand Data Type		DL06 Range	
	A	aaa	bbb
Inputs	X	0-777	—
Outputs	Y	0-777	—
Control Relays	C	0-1777	—
Stage Bits	S	0-1777	—
Timer Bits	T	0-377	—
Counter Bits	CT	0-177	—
Special Relays	SP	0-137 320-717	—
Global I/O	GX	0-3777	—
Constant	K	—	1-16

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On any time the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

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

In the following example, when X6 is on, the value formed by discrete locations X0-X3 is loaded into the accumulator using the Load Formatted instruction. The value in the accumulator is divided by the value formed by discrete location C0-C3 using the Divide Formatted instruction. The value in the lower four bits of the accumulator is copied to Y10-Y13 using the Out Formatted instruction.





Add Top of Stack (ADDS)

Add Top of Stack is a 32 bit instruction that adds the BCD value in the accumulator with the BCD value in the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack values are moved up one level.

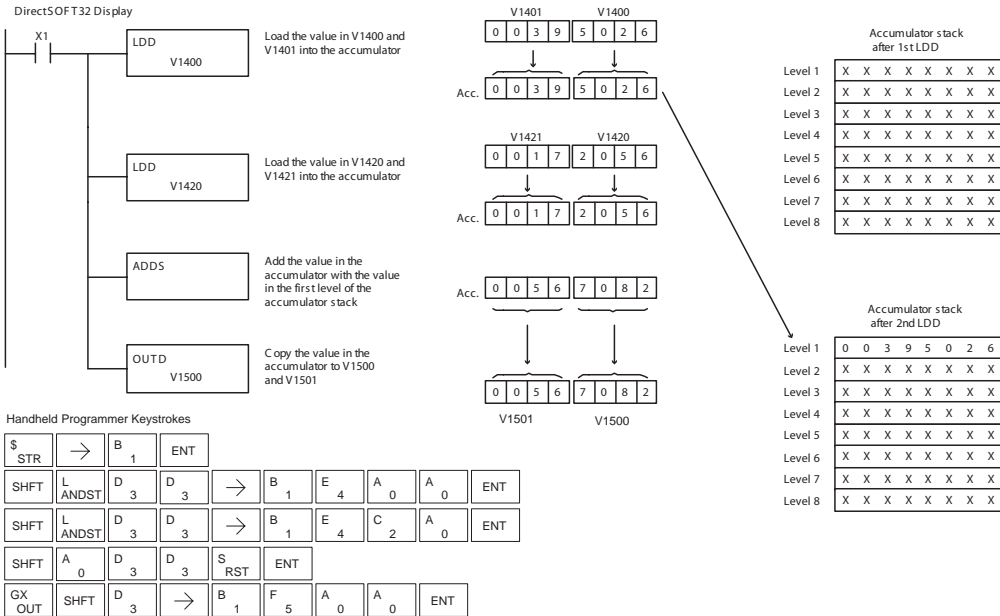
ADDS

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.



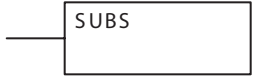
**NOTE:** Status flags are valid only until another instruction uses the same flag.

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)

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.

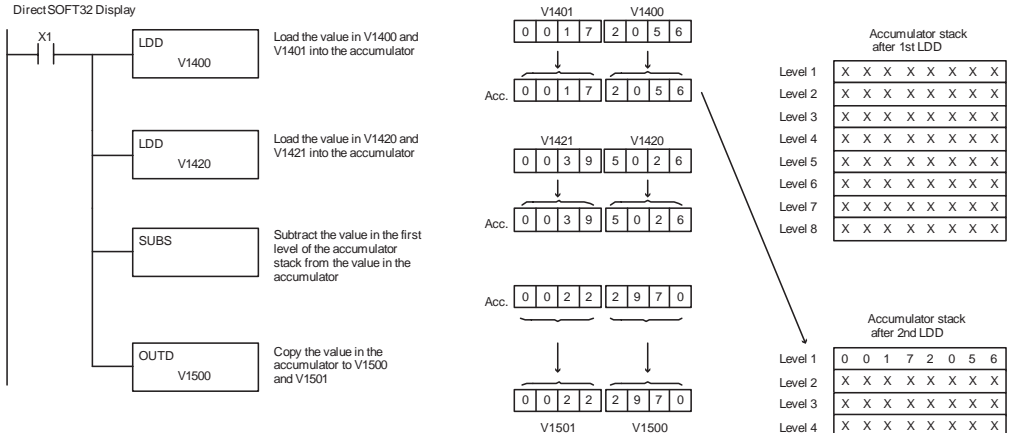


Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16 bit subtraction instruction results in a borrow
SP65	On when the 32 bit subtraction instruction results in a borrow
SP70	On any time the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

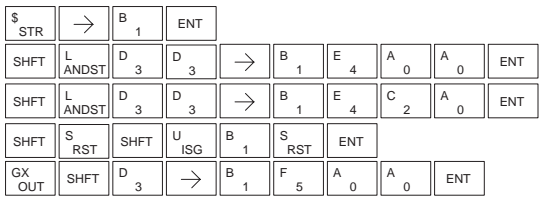


**NOTE:** Status flags are valid only until another instruction uses the same flag.

In the following example, when X1 is on, the value in V1400 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



Multiply Top of Stack (MULS)

Multiply Top of Stack is a 16 bit instruction that multiplies a 4-digit BCD value in the first level of the accumulator stack by a 4-digit BCD value in the accumulator. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack values are moved up one level.

MULS

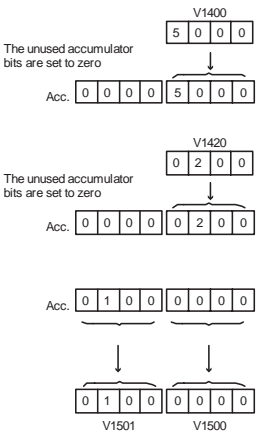
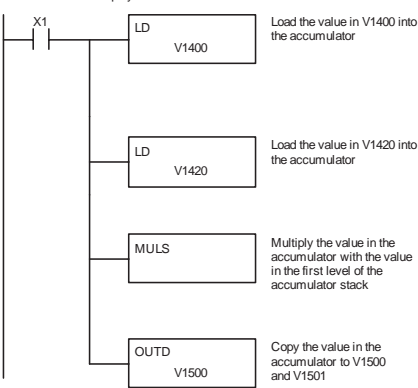
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On any time the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.



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

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The value in V1420 is loaded into the accumulator using the Load instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The BCD value in the first level of the accumulator stack is multiplied by the BCD value in the accumulator using the Multiply Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

DirectSOFT32 Display



Accumulator stack after 1st LDD

Level 1	X	X	X	X	X	X	X
Level 2	X	X	X	X	X	X	X
Level 3	X	X	X	X	X	X	X
Level 4	X	X	X	X	X	X	X
Level 5	X	X	X	X	X	X	X
Level 6	X	X	X	X	X	X	X
Level 7	X	X	X	X	X	X	X
Level 8	X	X	X	X	X	X	X

Accumulator stack after 2nd LDD

Level 1	0	0	0	0	5	0	0
Level 2	X	X	X	X	X	X	X
Level 3	X	X	X	X	X	X	X
Level 4	X	X	X	X	X	X	X
Level 5	X	X	X	X	X	X	X
Level 6	X	X	X	X	X	X	X
Level 7	X	X	X	X	X	X	X
Level 8	X	X	X	X	X	X	X

Handheld Programmer Keystrokes

\$	STR	→	B	1	ENT				
SHFT	L	ANDST	D	3	→	B	1	E	4
SHFT	L	ANDST	D	3	→	B	1	E	4
SHFT	M	ORST	U	ISG	L	ANDST	S	RST	ENT
GX	OUT	SHFT	D	3	→	B	1	F	5

### Divide by Top of Stack (DIVS)

Divide Top of Stack is a 32 bit instruction that divides the 8-digit BCD value in the accumulator by a 4-digit BCD value in the first level of the accumulator stack. The result resides in the accumulator and the remainder resides in the first level of the accumulator stack.

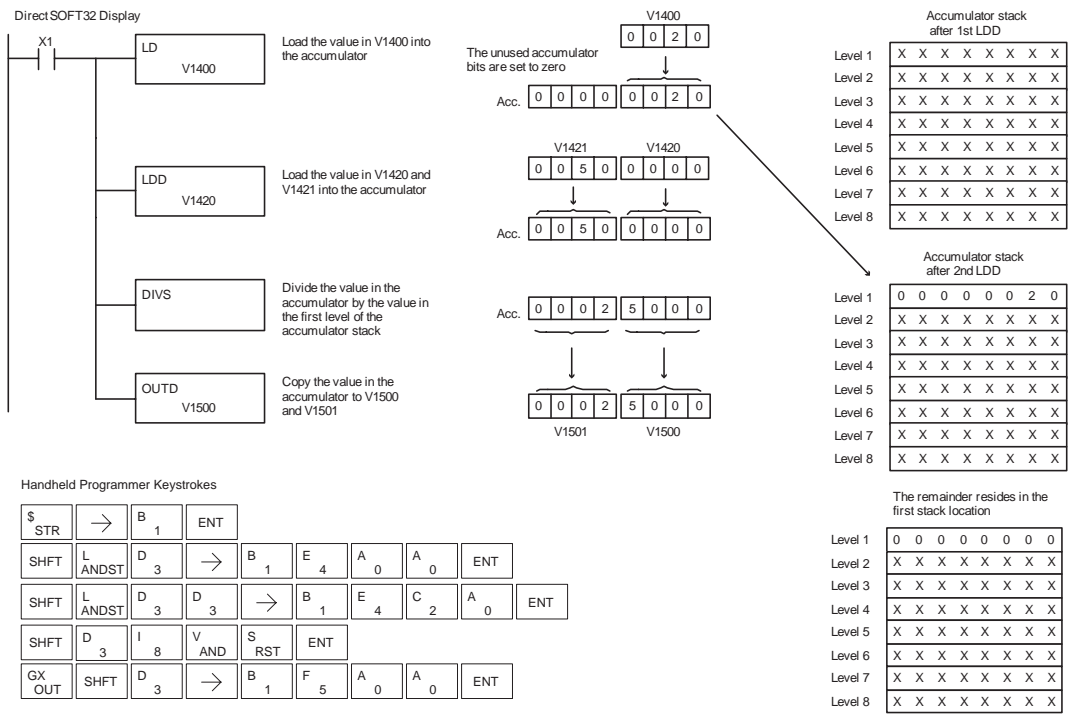
DIVS

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On any time the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.



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

In the following example, when X1 is on, the Load instruction loads the value in V1400 into the accumulator. The value in V1420 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The BCD value in the accumulator is divided by the BCD value in the first level of the accumulator stack using the Divide Stack instruction. The Out Double instruction copies the value in the accumulator to V1500 and V1501.



## Add Binary Top of Stack (ADDBS)

Add Binary Top of Stack instruction is a 32 bit instruction that adds the binary value in the accumulator with the binary value in the first level of the accumulator stack. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack values are moved up one level.

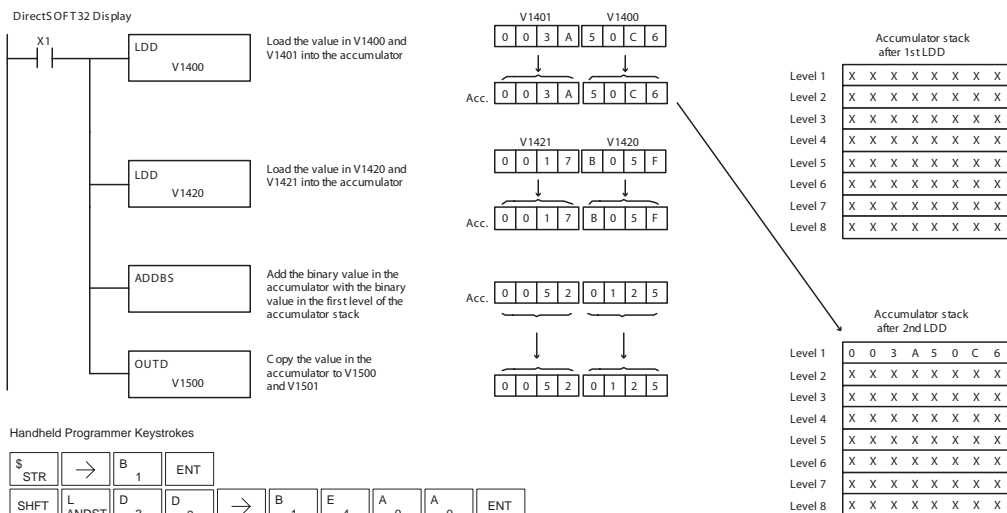
## ADDBS

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16 bit addition instruction results in a carry.
SP67	On when the 32 bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP73	On when a signed addition or subtraction results in an incorrect sign bit.



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

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction, pushing the value previously loaded in the accumulator onto the accumulator stack. The binary value in the first level of the accumulator stack is added with the binary value in the accumulator using the Add Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



## Subtract Binary Top of Stack (SUBBS)

Subtract Binary Top of Stack is a 32 bit instruction that subtracts the binary value in the first level of the accumulator stack from the binary value in the accumulator. The result resides in the accumulator. The value in the first level of the accumulator stack is removed and all stack locations are moved up one level.

SUBBS

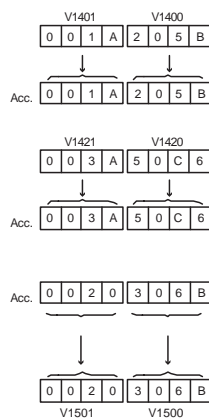
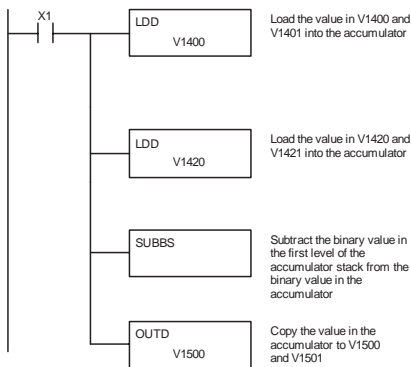
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16 bit subtraction instruction results in a borrow
SP65	On when the 32 bit subtraction instruction results in a borrow
SP70	On any time the value in the accumulator is negative.



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

In the following example, when X1 is on, the value in V1400 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.

DirectSOFT32 Display



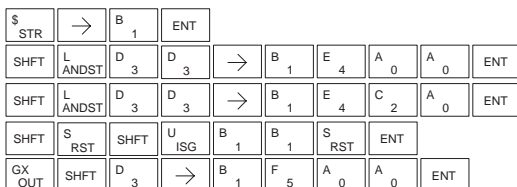
Accumulator stack after 1st LDD

Level 1	X X X X X X X X
Level 2	X X X X X X X X
Level 3	X X X X X X X X
Level 4	X X X X X X X X
Level 5	X X X X X X X X
Level 6	X X X X X X X X
Level 7	X X X X X X X X
Level 8	X X X X X X X X

Accumulator stack after 2nd LDD

Level 1	0 0 1 A 2 0 5 B
Level 2	X X X X X X X X
Level 3	X X X X X X X X
Level 4	X X X X X X X X
Level 5	X X X X X X X X
Level 6	X X X X X X X X
Level 7	X X X X X X X X
Level 8	X X X X X X X X

Handheld Programmer Keystrokes



Multiply Binary Top of Stack (MULBS)

Multiply Binary Top of Stack is a 16 bit instruction that multiplies the 16 bit binary value in the first level of the accumulator stack by the 16 bit binary value in the accumulator. The result resides in the accumulator and can be 32 bits (8 digits max.). The value in the first level of the accumulator stack is removed and all stack locations are moved up one level.

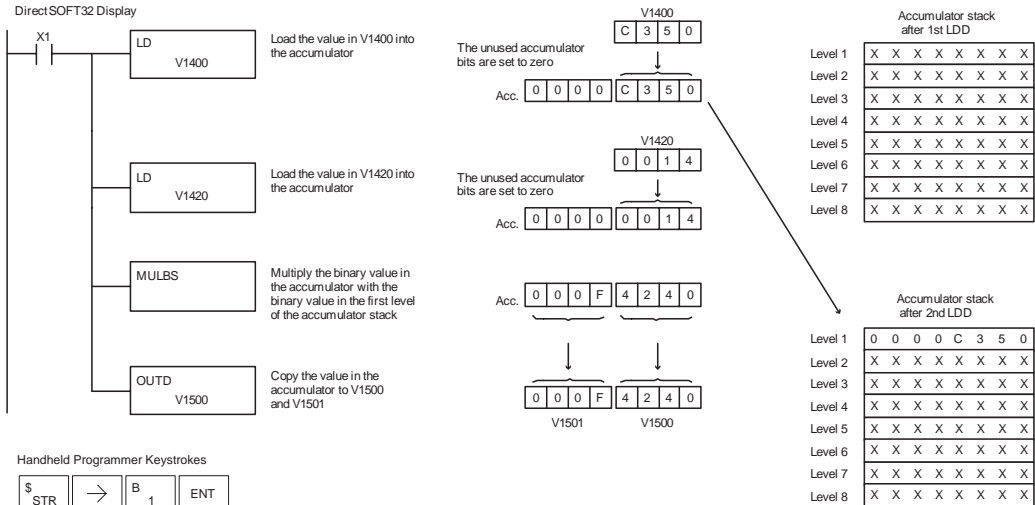
MULBS

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On any time the value in the accumulator is negative.



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

In the following example, when X1 is on, the Load instruction moves the value in V1400 into the accumulator. The value in V1420 is loaded into the accumulator using the Load instruction, pushing the value previously loaded in the accumulator onto the stack. The binary value in the accumulator stack's first level is multiplied by the binary value in the accumulator using the Multiply Binary Stack instruction. The Out Double instruction copies the value in the accumulator to V1500 and V1501.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	B 1	E 4	A 0	A 0	ENT	
SHFT	L ANDST	D 3	→	B 1	E 4	C 2	A 0	ENT	
SHFT	M ORST	U ISG	L ANDST	B 1	S RST	ENT			
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT	

## Divide Binary by Top OF Stack (DIVBS)

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.

DIVBS

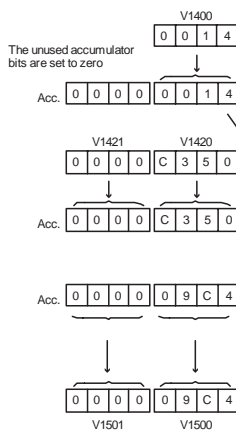
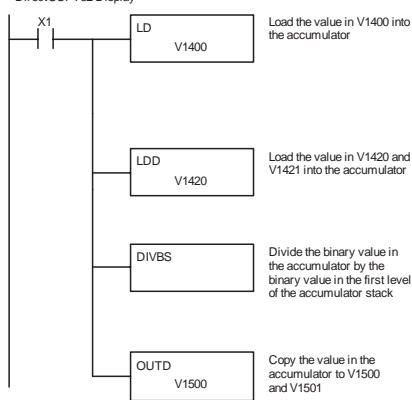
Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On any time the value in the accumulator is negative.



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

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The value in V1420 and V1421 is loaded into the accumulator using the Load Double instruction also, pushing the value previously loaded in the accumulator onto the accumulator stack. The binary value in the accumulator is divided by the binary value in the first level of the accumulator stack using the Divide Binary Stack instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.

DirectSOFT32 Display



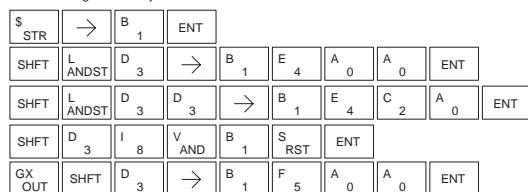
Accumulator stack after 1st LDD

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 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



The remainder resides in the first stack location

Level 1	0 0 0 0 0 0 0 0
Level 2	X X X X X X X X
Level 3	X X X X X X X X
Level 4	X X X X X X X X
Level 5	X X X X X X X X
Level 6	X X X X X X X X
Level 7	X X X X X X X X
Level 8	X X X X X X X X



# Transcendental Functions

The DL06 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
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.
SP72	On anytime the value in the accumulator is a valid floating point number
SP73	On anytime the value in the accumulator is negative.
SP75	On when a real number instruction is executed and a non-real number was encountered.

## Sine Real (SINR)

The Sine Real instruction takes the sine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

## Cosine Real (COSR)

The Cosine Real instruction takes the cosine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

## Tangent Real (TANR)

The Tangent Real instruction takes the tangent of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

## Arc Sine Real (ASINR)

The Arc Sine Real instruction takes the inverse sine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

SINR

COSR

SINR

ASINR

## Arc Cosine Real (ACOSR)

The Arc Cosine Real instruction takes the inverse cosine of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

COSR

## Arc Tangent Real (ATANR)

The Arc Tangent Real instruction takes the inverse tangent of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

ATANR

## Square Root Real (SQRTR)

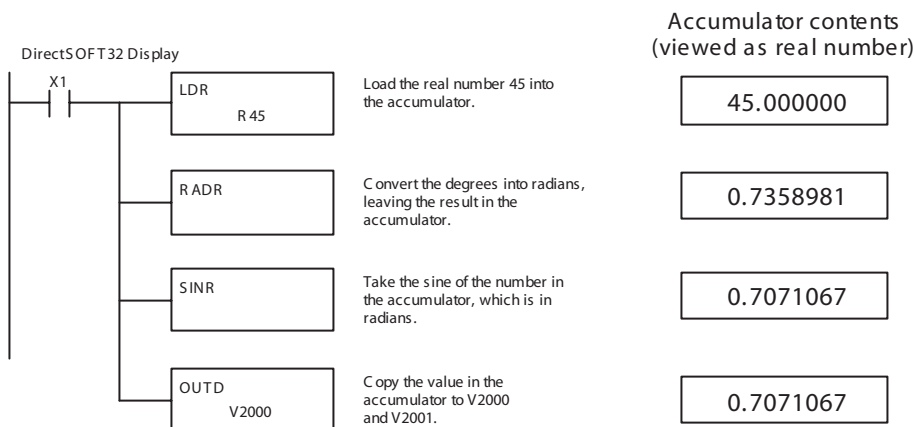
The Square Root Real instruction takes the square root of the real number stored in the accumulator. The result resides in the accumulator. Both the original number and the result are in IEEE 32-bit format.

SQRTR



**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 DirectSOFT32 for entering real numbers, using the LDR (Load Real) instruction.

# Bit Operation Instructions

## Sum (SUM)

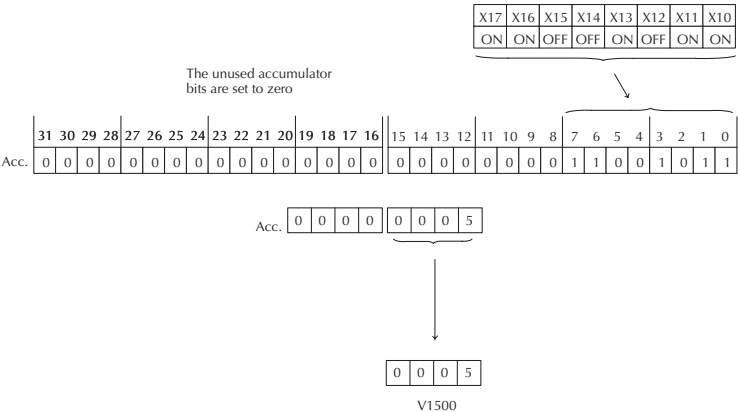
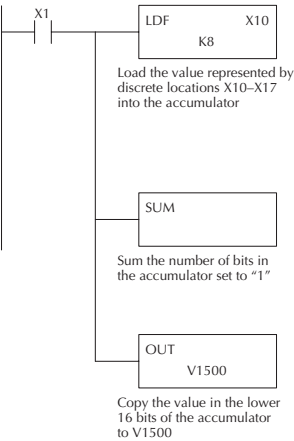
The Sum instruction counts number of bits that are set to “1” in the accumulator. The HEX result resides in the accumulator.

SUM

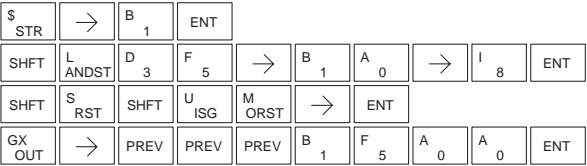
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.

In the following example, when X1 is on, the value formed by discrete locations X10–X17 is loaded into the accumulator using the Load Formatted instruction. The number of bits in the accumulator set to “1” is counted using the Sum instruction. The value in the accumulator is copied to V1500 using the Out instruction.

DirectSOFT32 Display



Handheld Programmer Keystrokes



## Shift Left (SHFL)

Shift Left is a 32 bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the left. The vacant positions are filled with zeros and the bits shifted out of the accumulator are discarded.

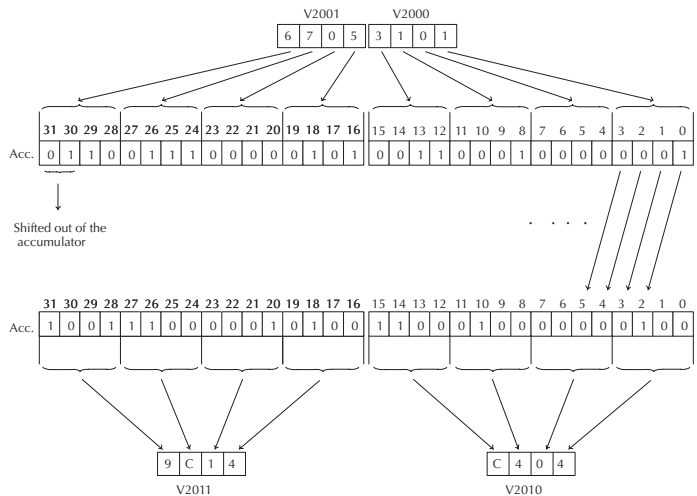
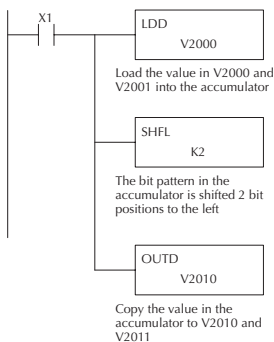
SHFL  
A aaa

Operand Data Type	DL06 Range
.....A	<b>aaa</b>
V memory .....V	See memory map
Constant .....K	1-32

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is shifted 2 bits to the left using the Shift Left instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.

DirectSOFT32



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	S RST	SHFT	H 7	F 5	L ANDST	→	C 2	ENT	
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

Shift Right (SHFR)

Shift Right is a 32 bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the right. The vacant positions are filled with zeros and the bits shifted out of the accumulator are lost.

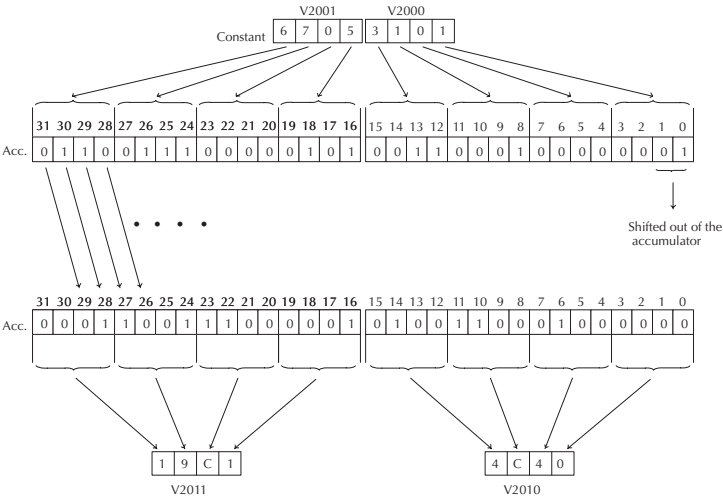
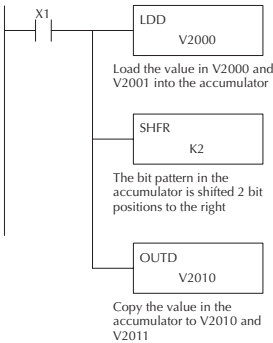
SHFR  
A aaa

Operand Data Type	DL06Range
..... A	<b>aaa</b>
V memory ..... V	See memory map
Constant ..... K	1-32

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is shifted 2 bits to the right using the Shift Right instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.

Direct SOFT32



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	S RST	SHFT	H 7	F 5	R ORN	→	C 2	ENT	
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

## Rotate Left (ROTL)

Rotate Left is a 32 bit instruction that rotates the bits in the accumulator a specified number (Aaaa) of places to the left.

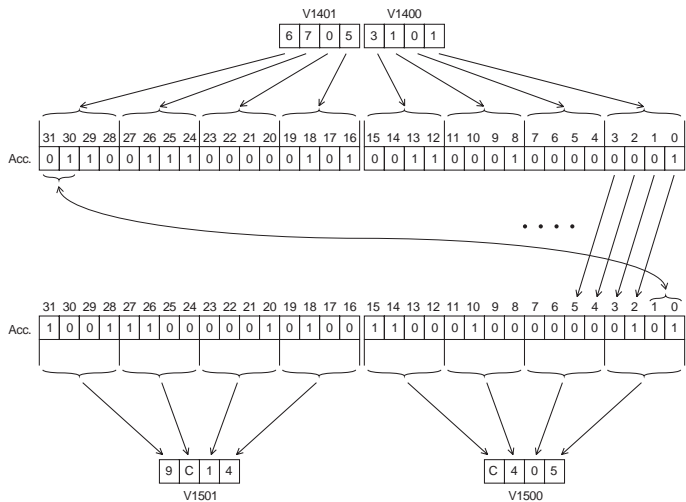
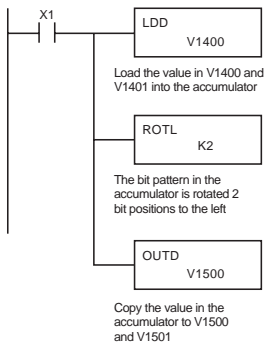
ROTL

Aaaa

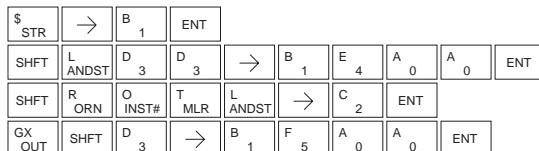
Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Constant ..... K	1-32

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.

DirectSOFT32 Display



Handheld Programmer Keystrokes



Rotate Right (ROTR)

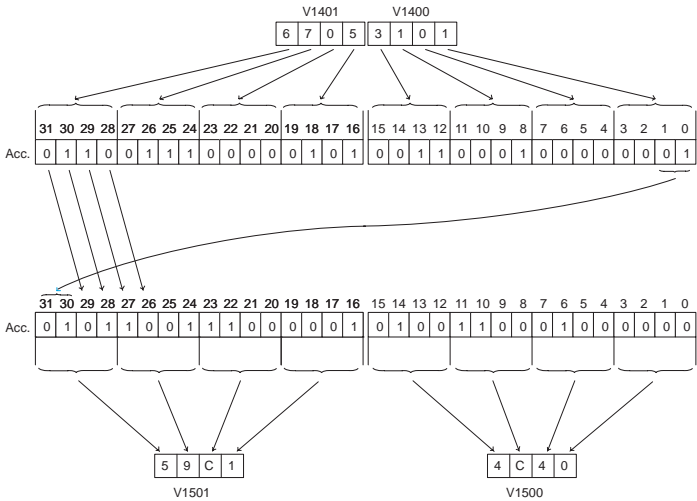
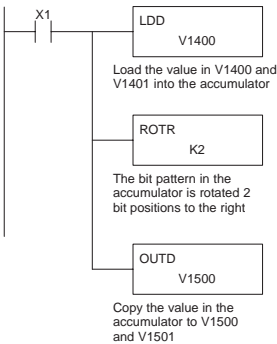
Rotate Right is a 32 bit instruction that rotates the bits in the accumulator a specified number (Aaaa) of places to the right.

ROTR  
A aaa

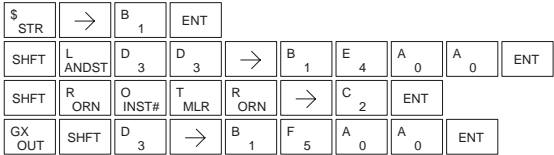
Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Constant ..... K	1-32

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.

Direct SOFT Display



Handheld Programmer Keystrokes



## Encode (ENCO)

The Encode instruction encodes the bit position in the accumulator having a value of 1, and returns the appropriate binary representation. If the most significant bit is set to 1 (Bit 31), the Encode instruction would place the value HEX 1F (decimal 31) in the accumulator. If the value to be encoded is 0000 or 0001, the instruction will place a zero in the accumulator. If the value to be encoded has more than one bit position set to a “1”, the least significant “1” will be encoded and SP53 will be set on.

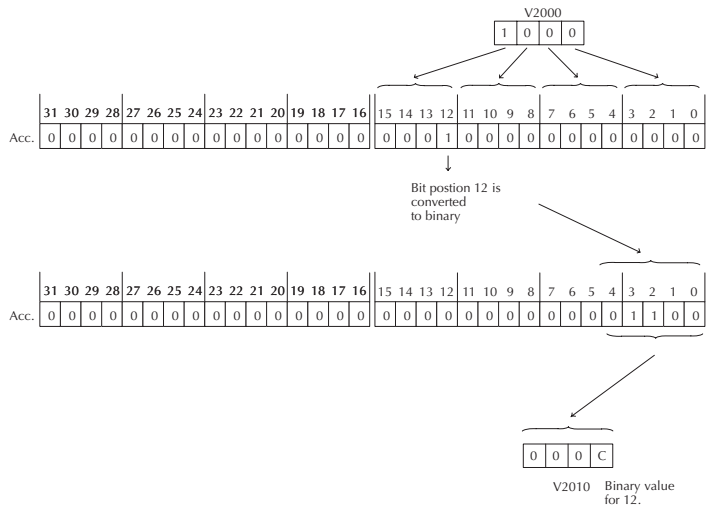
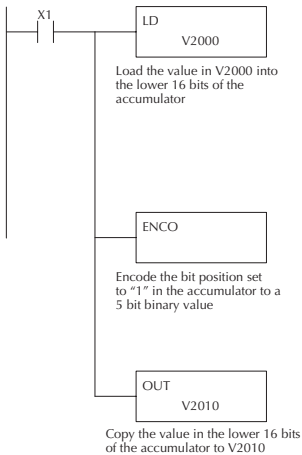
ENCO

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:** The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, The value in V2000 is loaded into the accumulator using the Load instruction. The bit position set to a “1” in the accumulator is encoded to the corresponding 5 bit binary value using the Encode instruction. The value in the lower 16 bits of the accumulator is copied to V2010 using the Out instruction.

Direct SOFT32



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT	
SHFT	E 4	N TMR	C 2	O INST#	ENT				
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT	

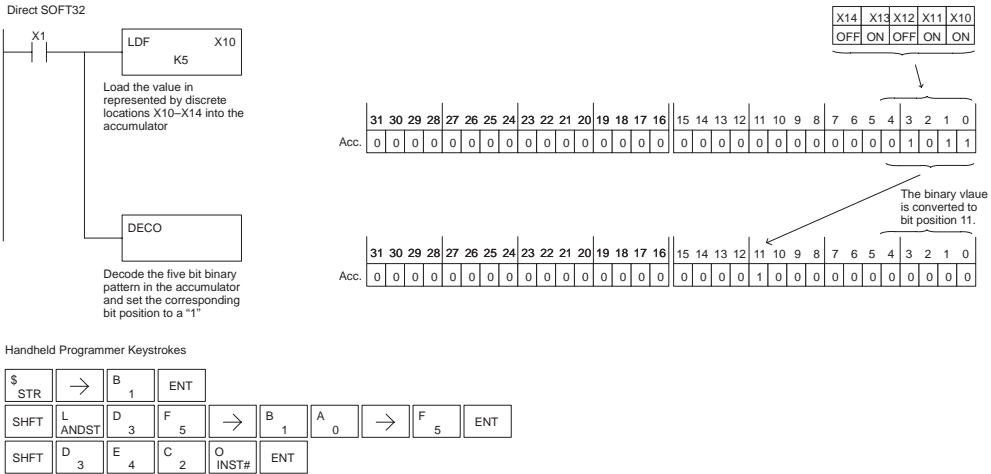


Decode (DECO)

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.

DECO

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.



# Number Conversion Instructions (Accumulator)

## Binary (BIN)

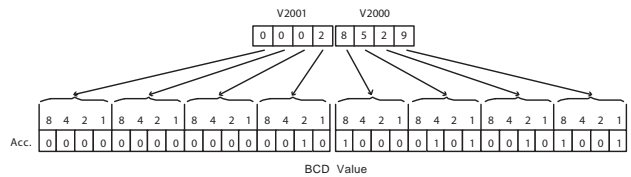
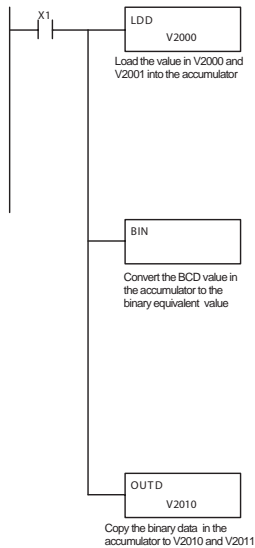
The Binary instruction converts a BCD value in the accumulator to the equivalent binary value. The result resides in the accumulator.

BIN

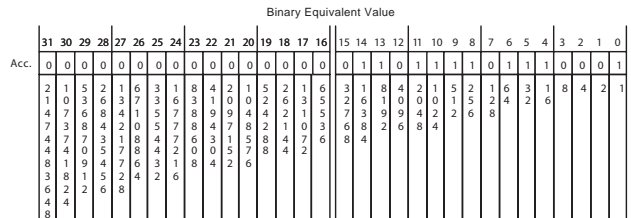
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

In the following example, when X1 is on, the value in V2000 and V2001 is loaded into the accumulator using the Load Double instruction. The BCD value in the accumulator is converted to the binary (HEX) equivalent using the BIN instruction. The binary value in the accumulator is copied to V2010 and V2011 using the Out Double instruction. (The handheld programmer will display the binary value in V2010 and V2011 as a HEX value.)

Direct5 OF T32

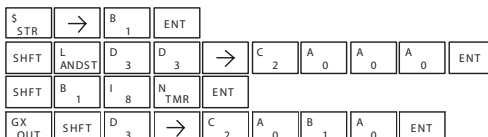


$$28529 = 16384 + 8192 + 2048 + 1024 + 512 + 256 + 64 + 32 + 16 + 1$$



The Binary (HEX) value copied to V2010

Handheld Programmer Keystrokes



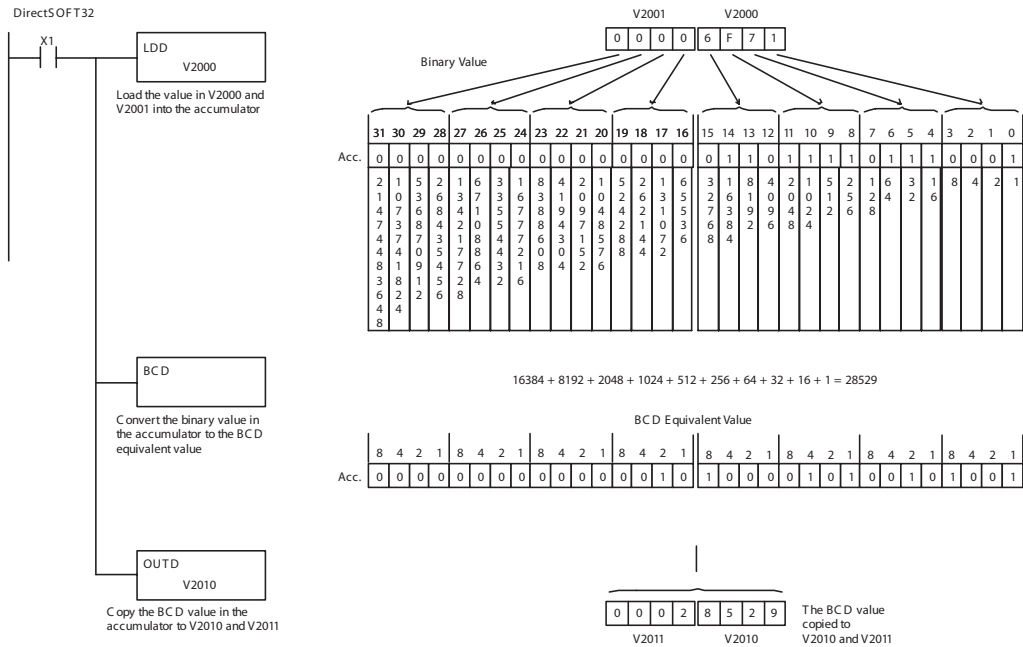
Binary Coded Decimal (BCD)

The Binary Coded Decimal instruction converts a binary value in the accumulator to the equivalent BCD value. The result resides in the accumulator.

BCD

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.

In the following example, when X1 is on, the binary (HEX) value in V2000 and V2001 is loaded into the accumulator using the Load Double instruction. The binary value in the accumulator is converted to the BCD equivalent value using the BCD instruction. The BCD value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	B 1	C 2	D 3	ENT					
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0		ENT



Ten's Complement (BCDCPL)

The Ten's Complement instruction takes the 10's complement (BCD) of the 8 digit accumulator. The result resides in the accumulator. The calculation for this instruction is :

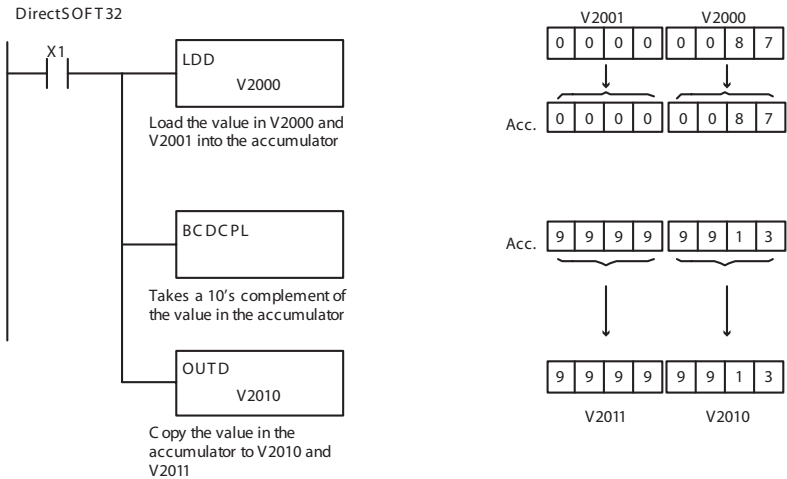
100000000

- accumulator value

10's complement value

BCDCPL

In the following example when X1 is on, the value in V2000 and V2001 is loaded into the accumulator. The 10's complement is taken for the 8 digit accumulator using the Ten's Complement instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT																	
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT											
SHFT	B 1	C 2	D 3	C 2	P CV	L ANDST	ENT													
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT												

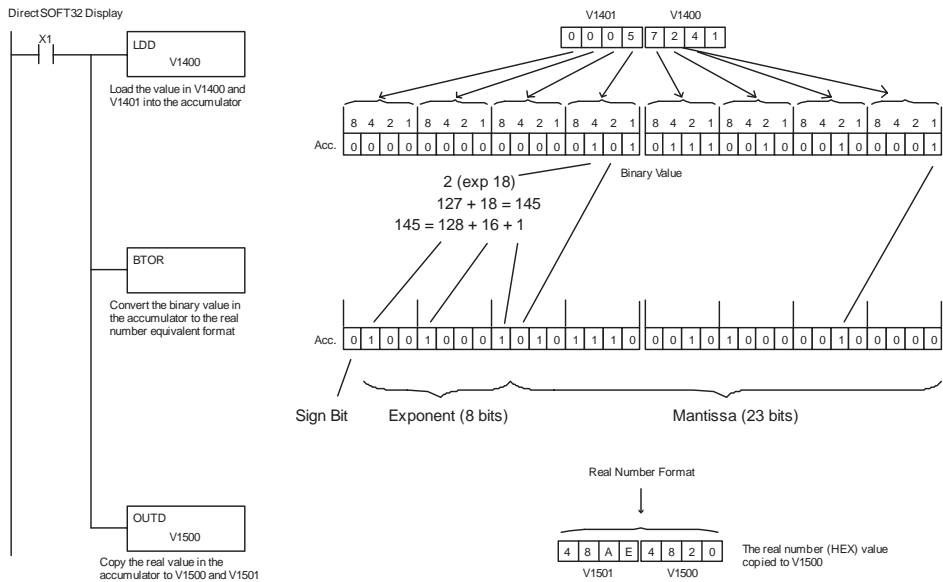
## Binary to Real Conversion (BTOR)

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.

BTOR

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	→	B 1	ENT
SHFT	L ANDST	D 3	D 3 → B 1 E 4 A 0 A 0 ENT
SHFT	B 1	T MLR	O INST# R ORN ENT
GX OUT	SHFT	D 3	→ B 1 F 5 A 0 A 0 ENT

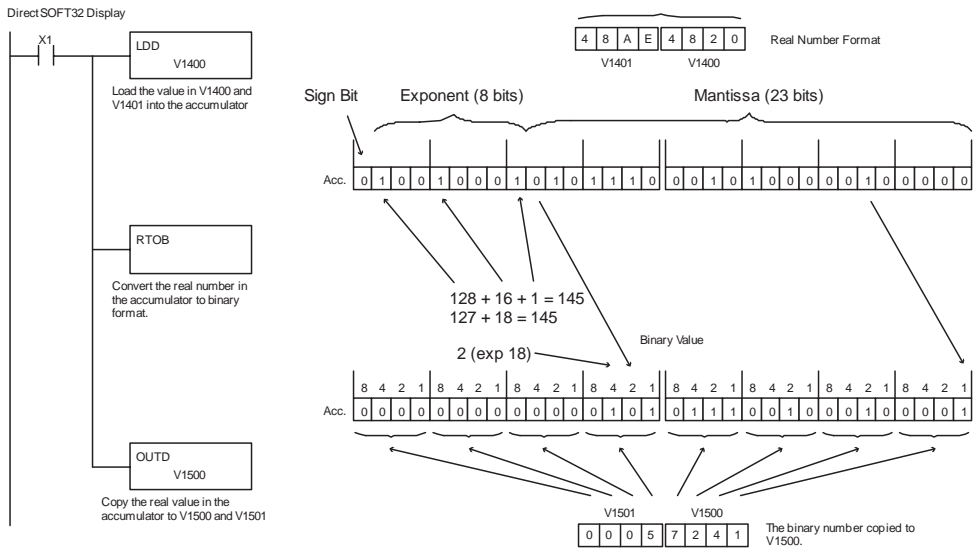
## Real to Binary Conversion (RTOB)

The Real-to-Binary instruction converts the real number in the accumulator to a binary value. The result resides in the accumulator. Both the binary and the real number may use all 32 bits of the accumulator.

RTOB

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP72	On anytime the value in the accumulator is a valid floating point number.
SP73	On when a signed addition or subtraction results in an incorrect sign bit.
SP75	On when a number cannot be converted to binary.

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	→	B 1	ENT							
SHFT	L ANDST	D 3	D 3	→	B 1	E 4	A 0	A 0	ENT	
SHFT	R ORN	T MLR	O INST#	B 1	ENT					
GX OUT	SHFT	D 3	→	B 1	F 5	A 0	A 0	ENT		

## Radian Real Conversion (RADR)

The Radian Real Conversion instruction converts the real degree value stored in the accumulator to the equivalent real number in radians. The result resides in the accumulator.



## Degree Real Conversion (DEGR)

The Degree Real instruction converts the degree real radian value stored in the accumulator to the equivalent real number in degrees. The result resides in the accumulator.



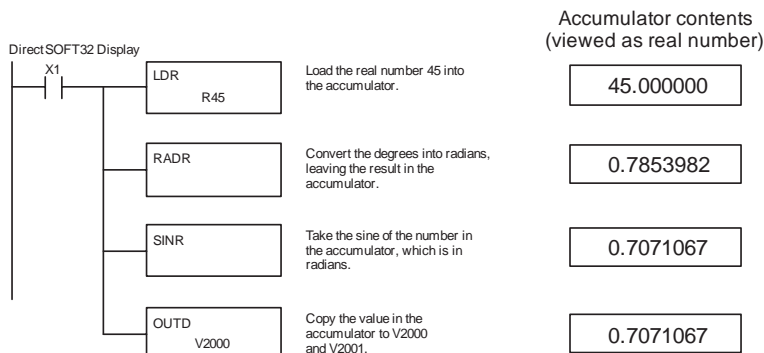
The two instructions described above convert real numbers into the accumulator from degree format to radian format, and visa-versa. In degree format, a circle contains 360 degrees. In radian format, a circle contains about 6.28 radians. These convert between both positive and negative real numbers, and for angles greater than a full circle. These functions are very useful when combined with the transcendental trigonometric functions (see the section on math instructions).

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
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 **DirectSOFT32** 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)

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.

ATH  
Vaaa

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.

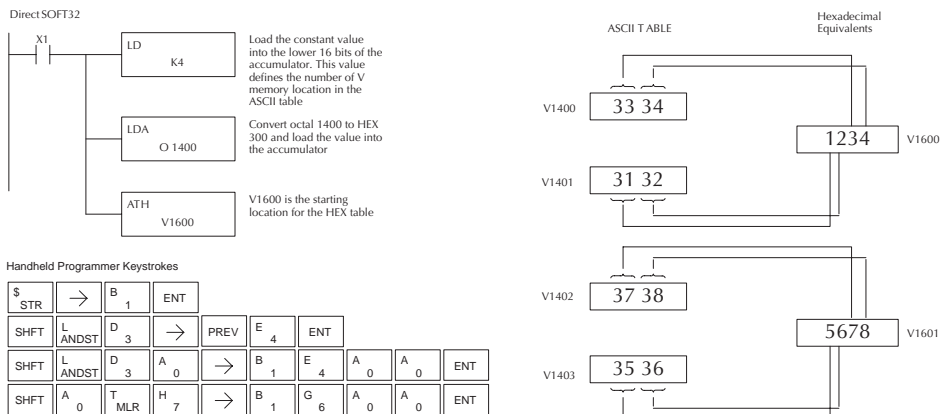
Step 3: — Specify the starting V memory location (Vaaa) for the HEX table in the ATH instruction.

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	DL06 Range
	<b>aaa</b>
V memory ..... V	See memory map
Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.

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)

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.

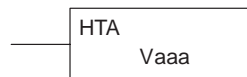
This means a HEX table of two V memory locations would require four V memory locations for the equivalent ASCII table. The function parameters are loaded into the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program a HEX to ASCII table 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.

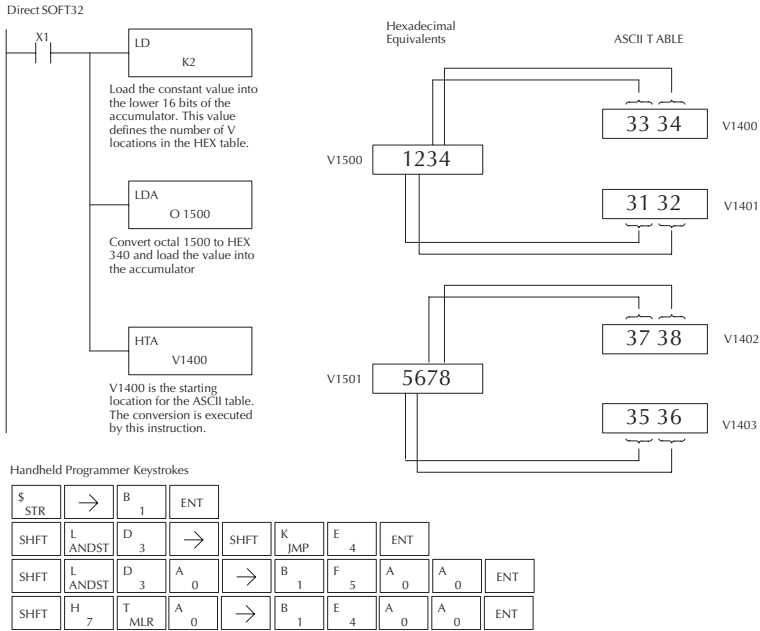
**Helpful Hint:** — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.



Operand Data Type	DL06 Range
	<b>aaa</b>
V memory ..... V	See memory map

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.

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



Gray Code (GRAY)

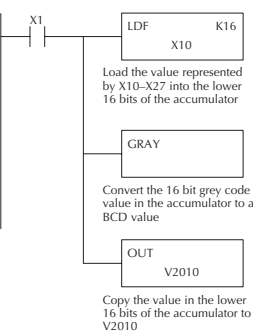
The Gray code instruction converts a 16 bit gray code value to a BCD value. The BCD conversion requires 10 bits of the accumulator. The upper 22 bits are set to “0”. This instruction is designed for use with devices (typically encoders) that use the grey code numbering scheme. The Gray Code instruction 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.

GRAY

In the following example, when X1 is ON the binary value represented by X10–X27 is loaded into the accumulator using the Load Formatted instruction. The gray code value in the accumulator is converted to BCD using the Gray Code instruction. The value in the lower 16 bits of the accumulator is copied to V2010.

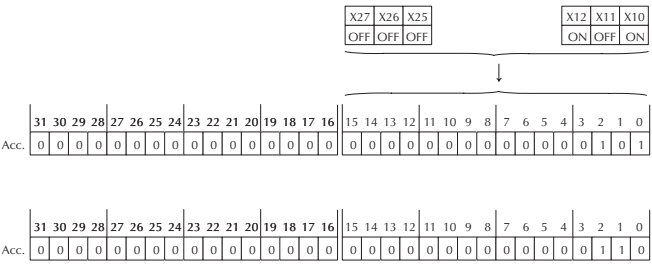
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.

DirectSOFT32



Handheld Programmer Keystrokes

\$	STR	→	B	1	ENT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
----	-----	---	---	---	-----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--



Gray Code	BCD
0000000000	0000
0000000001	0001
0000000011	0002
0000000010	0003
0000000110	0004
0000000111	0005
0000000101	0006
0000000100	0007
1000000001	1022
1000000000	1023

## Shuffle Digits (SFLDGT)

The Shuffle Digits instruction shuffles a maximum of 8 digits rearranging them in a specified order. This function requires parameters to be loaded into the first level of the accumulator stack and the accumulator with two additional instructions. Listed below are the steps necessary to use the shuffle digit function. The example on the following page shows a program for the Shuffle Digits function.

SFLDGT

Step 1: Load the value (digits) to be shuffled into the first level of the accumulator stack.

Step 2: Load the order that the digits will be shuffled to into the accumulator.

Step 3: Insert the SFLDGT instruction.



*Note: If the number used to specify the order contains a 0 or 9–F, the corresponding position will be set to 0.*

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.

## 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.

Digits to be shuffled (first stack location)

9	A	B	C	D	E	F	0
---	---	---	---	---	---	---	---

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

1	2	8	7	3	6	5	4
---	---	---	---	---	---	---	---

Specified order (accumulator)

Bit Positions    8   7   6   5   4   3   2   1

B	C	E	F	0	D	A	9
---	---	---	---	---	---	---	---

Result (accumulator)

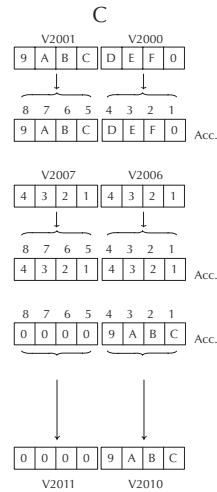
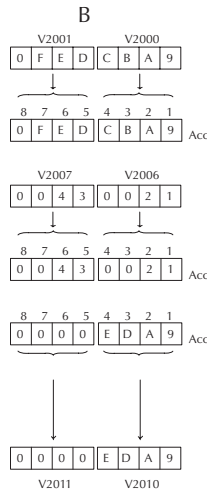
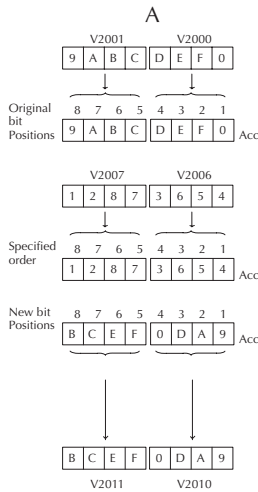
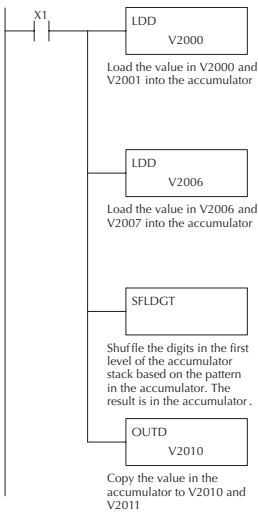
In the following example when X1 is on, The value in the first level of the accumulator stack will be reorganized in the order specified by the value in the accumulator.

Example A shows how the shuffle digits works when 0 or 9–F is not used when specifying the order the digits are to be shuffled. Also, there are no duplicate numbers in the specified order.

Example B shows how the shuffle digits works when a 0 or 9–F is used when specifying the order the digits are to be shuffled. Notice when the Shuffle Digits instruction is executed, the bit positions in the first stack location that had a corresponding 0 or 9–F in the accumulator (order specified) are set to “0”.

Example C shows how the shuffle digits works when duplicate numbers are used specifying the order the digits are to be shuffled. Notice when the Shuffle Digits instruction is executed, the most significant duplicate number in the order specified is used in the result.

Direct SOFT32



Handheld Programmer Keystrokes

S	→	B	ENT
SHFT	L	D	→
SHFT	L	D	→
SHFT	S	SHFT	F
GX	SHFT	D	→
OUT	SHFT	D	→
STR	→	B	ENT
SHFT	L	D	→
SHFT	L	D	→
SHFT	S	SHFT	F
GX	SHFT	D	→
OUT	SHFT	D	→

## Table Instructions

### Move (MOV)

The Move instruction moves the values from a V memory table to another V memory table the same length. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Move function.



- Step 1 Load the number of V memory locations to be moved into the first level of the accumulator stack. This parameter is a HEX value (K40 max, 100 octal).
- Step 2 Load the starting V memory location for the locations to be moved into the accumulator. This parameter is a HEX value.
- Step 3 Insert the MOVE instruction which specifies starting V memory location (Vaaa) for the destination table.

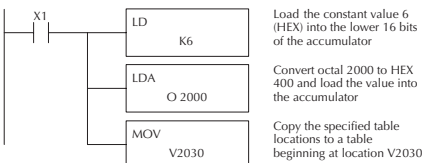
Helpful Hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type	DL06 Range
	<b>aaa</b>
V memory ..... V	See memory map
Pointer ..... P	See memory map

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.

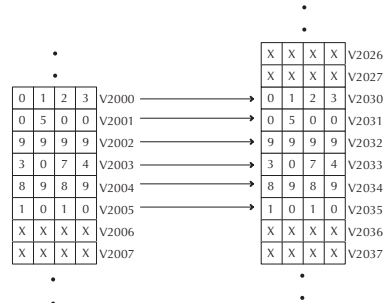
In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 2000 (V2000), the starting location for the source table is loaded into the accumulator. The destination table location (V2030) is specified in the Move instruction.

Direct SOFT32



Handheld Programmer Keystrokes

\$	STR	→	B	1	ENT
SHIFT	L	ANDST	D	3	→
SHIFT	L	ANDST	D	3	A
SHIFT	M	ORST	O	INST#	V
				AND	→
			C	2	A
			A	0	D
			A	0	A
					ENT

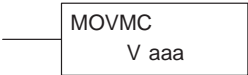




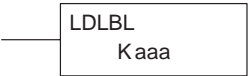
Move Memory Cartridge (MOVMC)

Load Label (LDLBL)

The Move Memory Cartridge and the Load Label instructions are used to copy data from program ladder memory to V memory. The Load Label instruction is used with the MOVMC instruction when copying data *from* program ladder memory to V memory.



To copy data from the program ladder memory to V memory, the function parameters are loaded into the first two levels of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Move Memory Cartridge and Load Label functions.

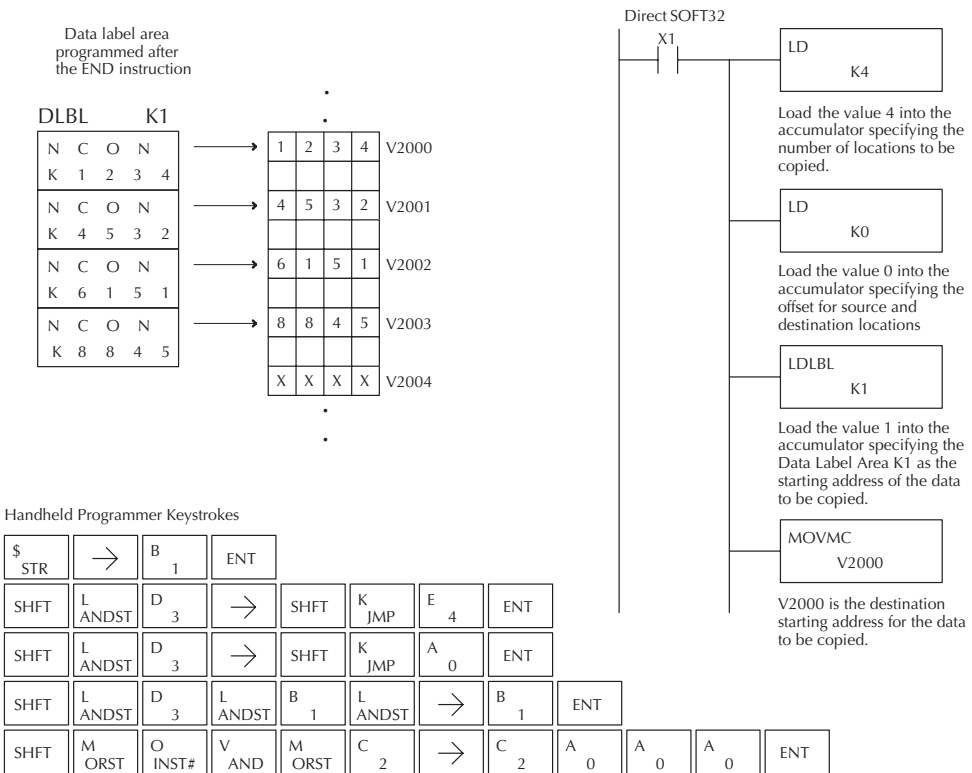


- Step 1: Load the number of words to be copied into the second level of the accumulator stack.
- Step 2: Load the offset for the data label area in ladder memory and the beginning of the V memory block into the first level of the stack.
- Step 3: Load the *source data label* (LDLBL Kaaa) into the accumulator when copying data from ladder memory to V memory. This is the source location of the value.
- Step 4: Insert the MOVMC instruction which specifies destination in V-memory (Vaaa). This is the copy destination.

Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map

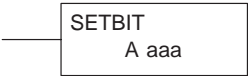
## Copy Data From a Data Label Area to V Memory

In the example below, data is copied from a Data Label Area to V memory. When X1 is on, the constant value (K4) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the second stack location after the next Load and Load Label (LDLBL) instructions are executed. The constant value (K0) is loaded into the accumulator, specifying the offset for the source and destination data. It is placed in the first stack location after the LDLBL instruction is executed. The source address where data is being copied from is loaded into the accumulator using the LDLBL instruction. The MOVMC instruction specifies the destination starting location and executes the copying of data from the Data Label Area to V memory.



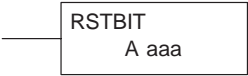
SETBIT

The Set Bit instruction sets a single bit to one within a range of V-memory locations.



RSTBIT

The Reset Bit instruction resets a single bit to zero within a range of V-memory locations.



The following description applies to both the Set Bit and Reset Bit table instructions.

Step 1: Load the length of the table (number of V memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2: Load the starting V memory location for the table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.

Step 3: Insert the Set Bit or Reset Bit instruction. This specifies the reference for the bit number of the bit you want to set or reset. The bit number is in octal, and the first bit in the table is number "0".

Helpful hint: — Remember that each V memory location contains 16 bits. So, the bits of the first word of the table are numbered from 0 to 17 octal. For example, if the table length is six words, then 6 words = (6 x 16) bits, = 96 bits (decimal), or 140 octal. The permissible range of bit reference numbers would be 0 to 137 octal. SP 53 will be set if the bit specified is outside the range of the table.

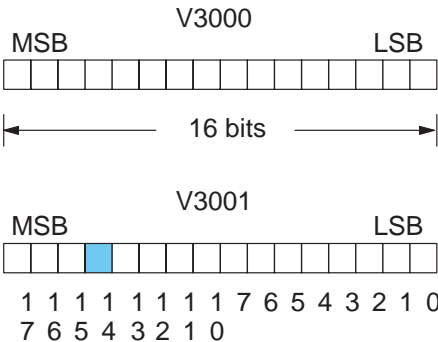
Operand Data Type	DL06 Range
	<b>aaa</b>
Vmemory ..... V	See Memory Map

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



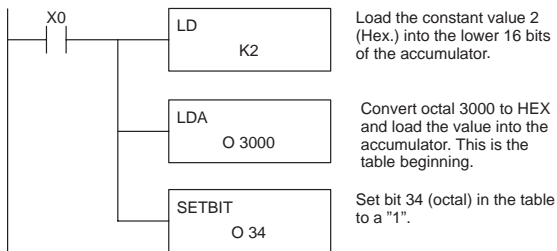
*NOTE: Status flags are only valid until the end of the scan or until another instruction that uses the same flag is executed.*

For example, suppose we have a table starting at V3000 that is two words long, as shown to the right. Each word in the table contains 16 bits, or 0 to 17 in octal. To set bit 12 in the second word, we use its octal reference (bit 14). Then we compute the bit's octal address from the start of the table, so  $17 + 14 = 34$  octal. The following program shows how to set the bit as shown to a "1".

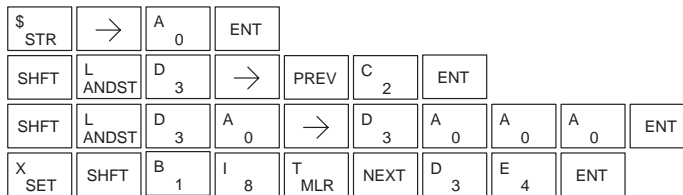


In this ladder example, we will use input X0 to trigger the Set Bit operation. First, we will load the table length (2 words) into the accumulator stack. Next, we load the starting address into the accumulator. Since V3000 is an octal number we have to convert it to hex by using the LDA command. Finally, we use the Set Bit (or Reset Bit) instruction and specify the octal address of the bit (bit 34), referenced from the table.

Direct SOFT Display32



Handheld Programmer Keystrokes



Fill (FILL)

The Fill instruction fills a table of up to 255 V memory locations with a value (Aaaa), which is either a V memory location or a 4-digit constant. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Fill function.



Step 1:— Load the number of V memory locations to be filled into the first level of the accumulator stack. This parameter must be a HEX value, 0–FF.

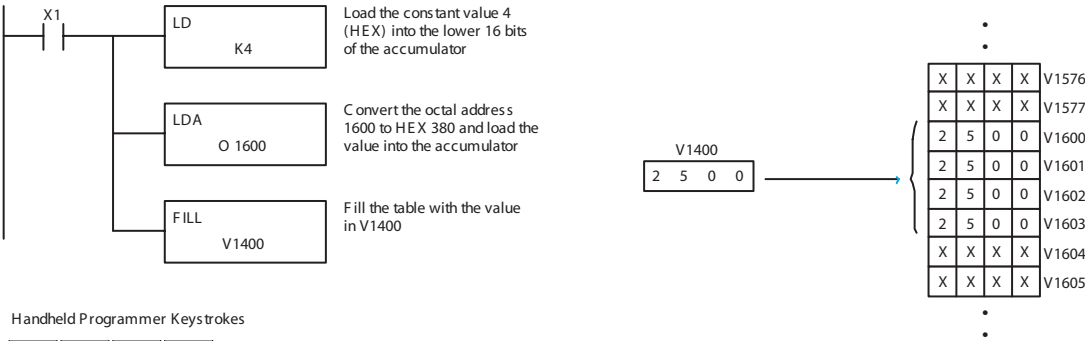
Step 2:— Load the starting V memory location for the table into the accumulator. This parameter must be a HEX value.

Step 3:— Insert the Fill 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.

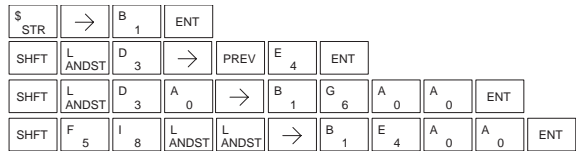
Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0–FF

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.

DirectS OFT32 Display



Handheld Programmer Keystrokes



## Find (FIND)

The Find instruction is used to search for a specified value in a V memory table of up to 255 locations. The function parameters are loaded into the first and second levels of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Find function.



Step 1: Load the length of the table (number of V memory locations) into the second level of the accumulator stack. This parameter must be a HEX value, 0–FF.

Step 2: Load the starting V memory location for the table into the first level of the accumulator stack. This parameter must be a HEX value.

Step 3: Load the offset from the starting location to begin the search. This parameter must be a HEX value.

Step 4: Insert the Find instruction which specifies the first value to be found in the table.

Results:— The offset from the starting address to the first V memory location which contains the search value is returned to the accumulator. SP53 will be set on if an address outside the table is specified in the offset or the value is not found. If the value is not found 0 will be returned in the accumulator.

Helpful Hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Constant ..... K	0–FFFF

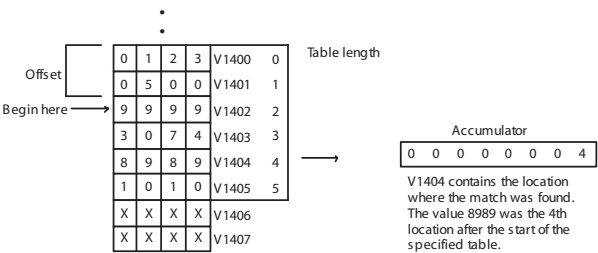
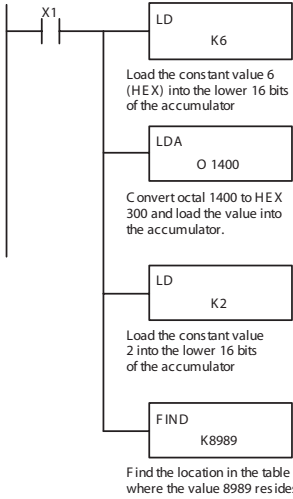
Discrete Bit Flags	Description
SP53	On if there is no value in the table that is equal to the search value.



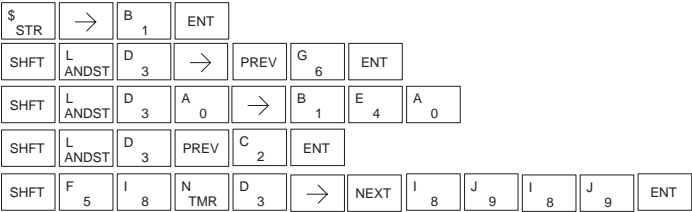
**NOTE:** Status flags are only valid until another instruction that uses the same flags is executed. The pointer for this instruction starts at 0 and resides in the accumulator.

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the 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.

DirectSOFT 32 Display



Handheld Programmer Keystrokes



Find Greater Than (FDGT)

The Find Greater Than instruction is used to search for the first occurrence of a value in a V memory table that is greater than the specified value (Aaaa), which can be either a V memory location or a 4-digit constant. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Find Greater Than function.



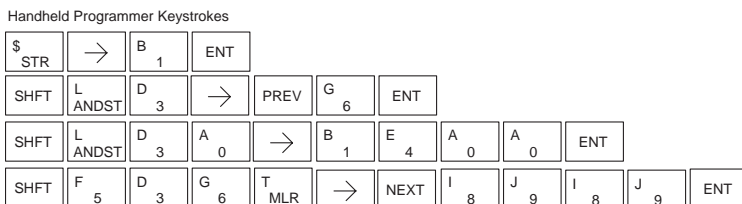
- Step 1: Load the length of the table (up to 255 locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0–FF.
- Step 2: Load the starting V memory location for the table into the accumulator. This parameter must be a HEX value.
- Step 3: Insert the FDGT instructions 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.

Helpful Hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

**Note:** This instruction does not have an offset, such as the one required for the FIND instruction.



Discrete Bit Flags	Description
SP53	On if there is no value in the table that is greater than the search value.





## Table to Destination (TTD)

The Table To Destination instruction moves a value from a V memory table to a V memory location and increments the table pointer by 1. The first V memory location in the table contains the table pointer which indicates the next location in the table to be moved. The instruction will be executed once per scan provided the input remains on. The table pointer will reset to 1 when the value equals the last location in the table. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Table To Destination function.



- Step 1: Load the length of the data table (number of V memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.
- Step 2: Load the starting V memory location for the table into the accumulator. (Remember, the starting location of the table is used as the table pointer.) This parameter must be a HEX value.
- Step 3: Insert the TTD instruction which specifies destination V memory location (Vaaa).

Helpful Hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful Hint:— The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful Hint: — The pointer location should be set to the value where the table operation will begin. The special relay SP0 or a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

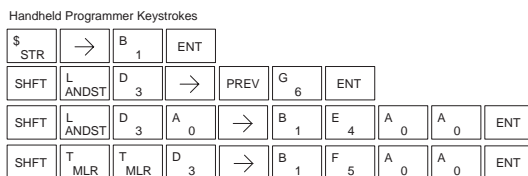
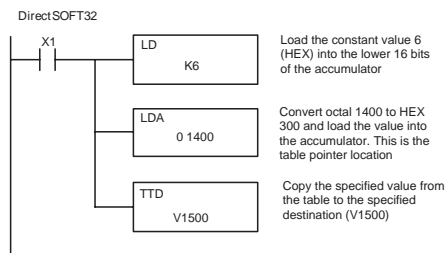
Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map

Discrete Bit Flags	Description
SP56	On when the table pointer equals the table length.



*NOTE: Status flags (SPs) are only valid until: — another instruction that uses the same flag is executed, or — the end of the scan The pointer for this instruction starts at 0 and resets when the table length is reached. At first glance it may appear that the pointer should reset to 0. However, it resets to 1, not 0.*

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400) is the starting location for the source table and is loaded into the accumulator. Remember, V1400 is used as the pointer location, and is not actually part of the table data source. The destination location (V1500) is specified in the Table to Destination instruction. The table pointer (V1400 in this case) will be increased by “1” after each execution of the TTD instruction.



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.

Table

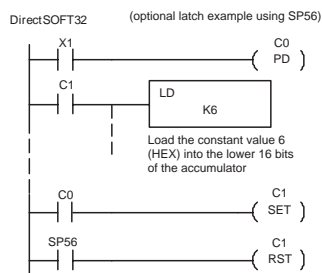
V1401	0	5	0	0	0 6
V1402	9	9	9	9	1
V1403	3	0	7	4	2
V1404	8	9	8	9	3
V1405	1	0	1	0	4
V1406	2	0	4	6	5
V1407	X	X	X	X	

Table Pointer

0	0	0	0	V1400
---	---	---	---	-------

Destination

X	X	X	X	V1500
---	---	---	---	-------

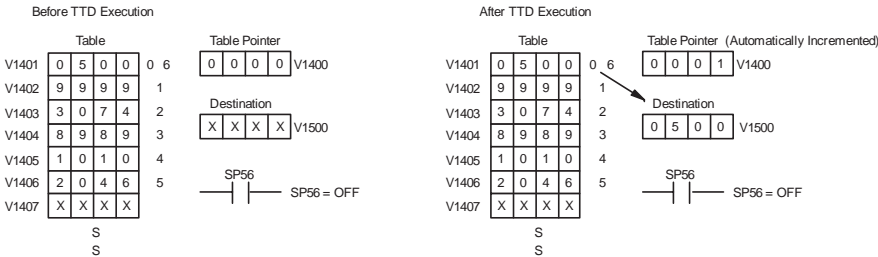


Since Special Relays are reset at the end of the scan, this latch must follow the TTD instruction in the program

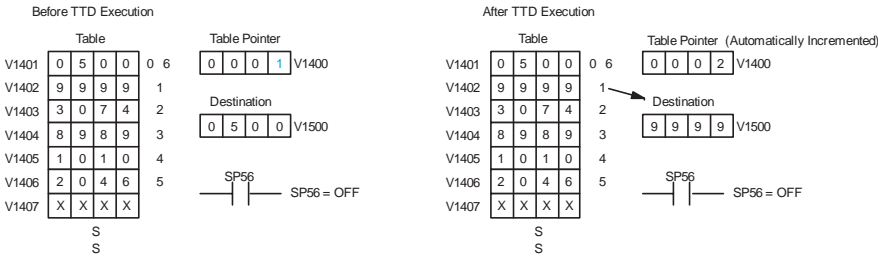
# Chapter 5: Standard RLL Instructions - Table Instructions

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.

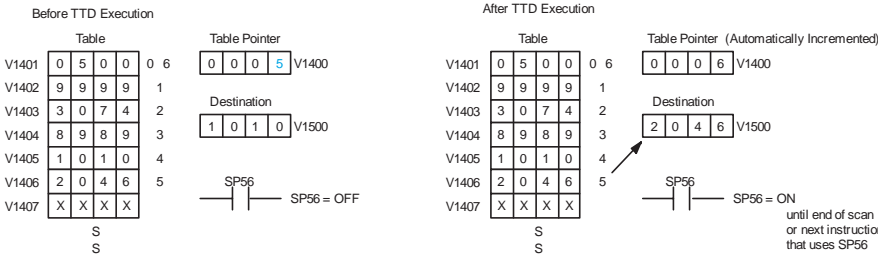
## Scan N



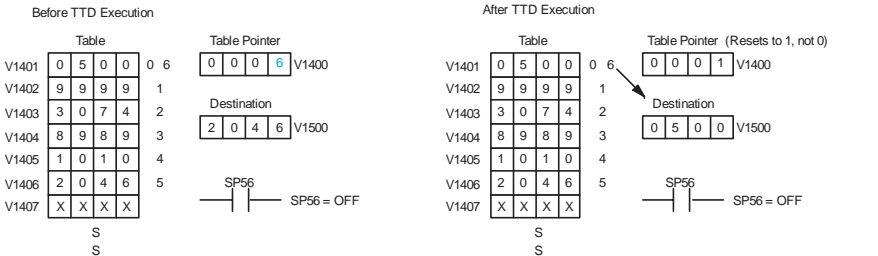
## Scan N+1



## Scan N+5



## Scan N+6



## Remove from Bottom (RFB)

The Remove From Bottom instruction moves a value from the bottom of a V memory table to a V memory location and decrements a table pointer by 1. The first V memory location in the table contains the table pointer which indicates the next location in the table to be moved. The instruction will be executed once per scan provided the input remains on. The instruction will stop operation when the pointer equals 0. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Remove From Bottom function.



Step 1:— Load the length of the table (number of V memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.

Step 2:— Load the starting V memory location for the table into the accumulator. (Remember, the starting location of the table blank is used as the table pointer.) This parameter must be a HEX value.

Step 3:— Insert the RFB 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.

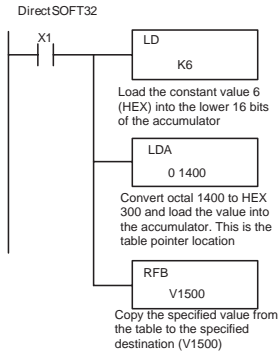
Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map

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.



Handheld Programmer Keystrokes

\$	STR	→	B	1	ENT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										</
----	-----	---	---	---	-----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

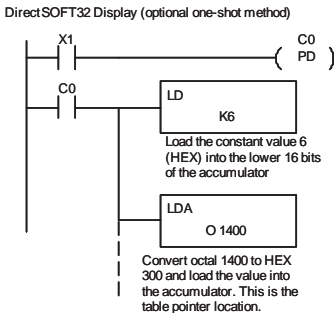
It is important to understand how the table locations are numbered. If you examine the example table, you'll notice that the first data location, V1401, will be used when the pointer is equal to one. The second data location, V1402, will be used when the pointer is equal to two, etc.

Table					
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	
•					

Table Pointer					
0	0	0	0	0	V1400

Destination					
X	X	X	X	X	V1500

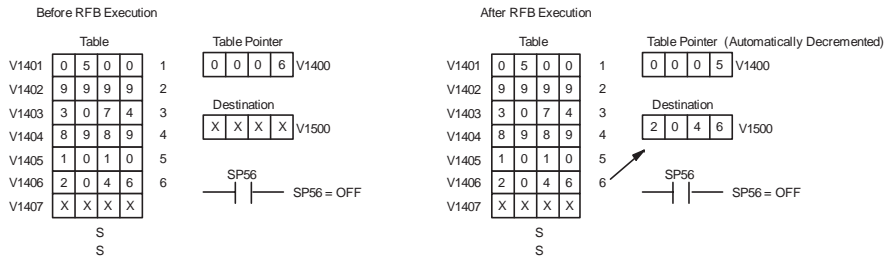
Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer decrements automatically, the table would cycle through the locations very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to remove one value each time the input contact transitions from low to high.



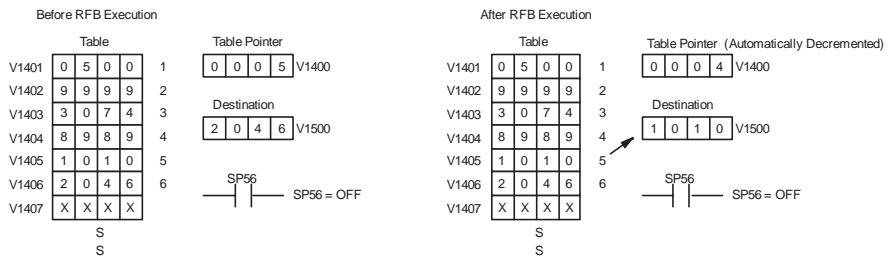
The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically decrements from 6 – 0. Also, notice how SP56 is only on until the end of the scan.

## Example of Execution

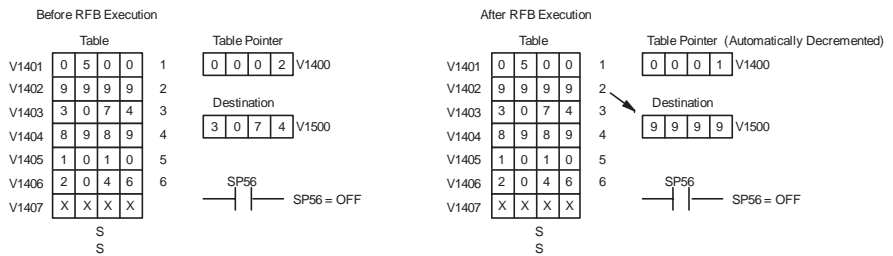
### Scan N



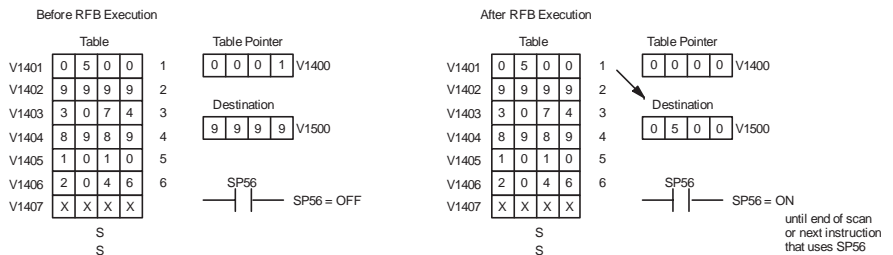
### Scan N+1



### Scan N+4



### Scan N+5



Source to Table (STT)

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	DL06 Range
	<b>aaa</b>
V memory ..... V	See memory map

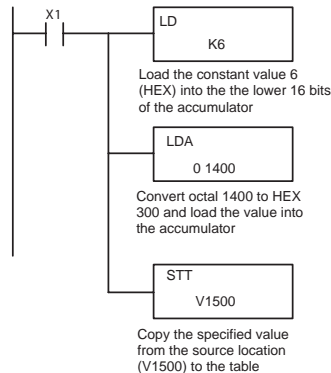
Discrete Bit Flags	Description
SP56	On when the table pointer equals the table length.



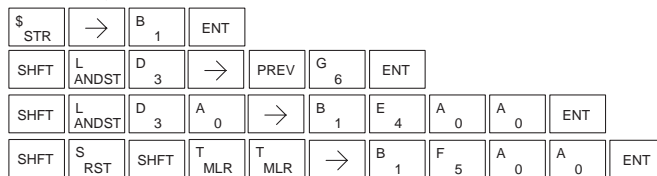
*NOTE: Status flags (SPs) are only valid until: — another instruction that uses the same flag is executed, or — the end of the scan The pointer for this instruction starts at 0 and resets to 1 automatically when the table length is reached.*

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 1400 (V1400), which is the starting location for the destination table and table pointer, is loaded into the accumulator. The data source location (V1500) is specified in the Source to Table instruction. The table pointer will be increased by “1” after each time the instruction is executed.

DirectSOF T32



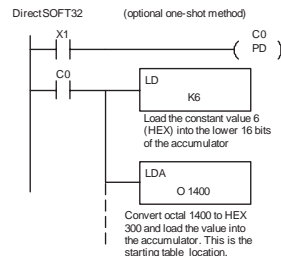
Handheld Programmer Keystrokes



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.

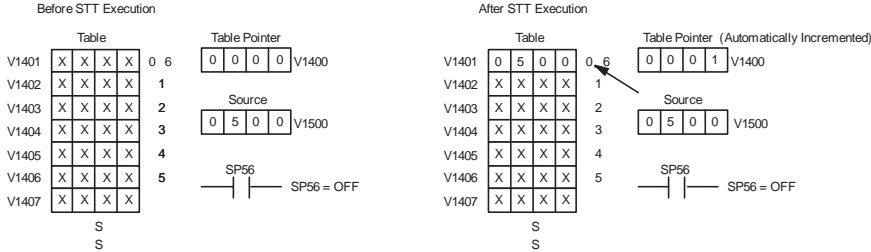
	Table					Table Pointer
V1401	X	X	X	X	0 6	0 0 0 0 V1400
V1402	X	X	X	X	1	
V1403	X	X	X	X	2	
V1404	X	X	X	X	3	Data Source
V1405	X	X	X	X	4	0 5 0 0 V1500
V1406	X	X	X	X	5	
V1407	X	X	X	X		



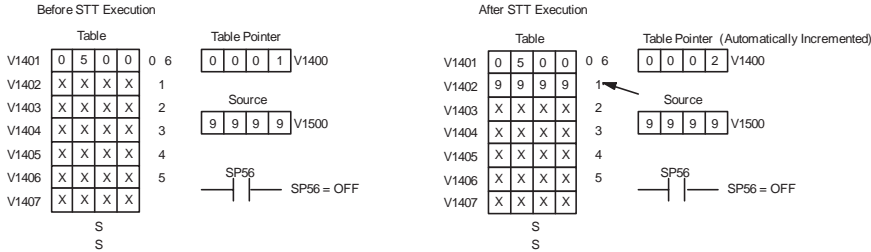


The following diagram shows the scan-by-scan results of the execution for our example program. Notice how the pointer automatically cycles from 0 – 6, and then starts over at 1 instead of 0. Also, notice how SP56 is affected by the execution. Although our example does not show it, we are assuming that 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.

## Scan N

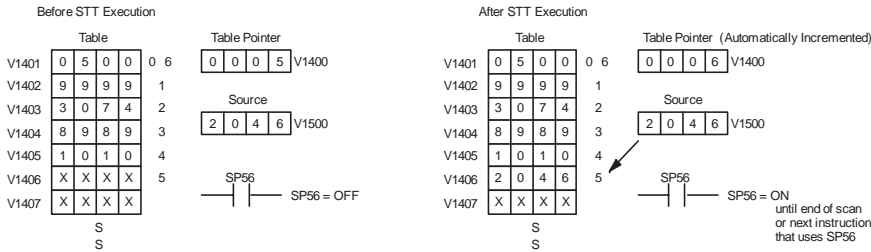


## Scan N+1

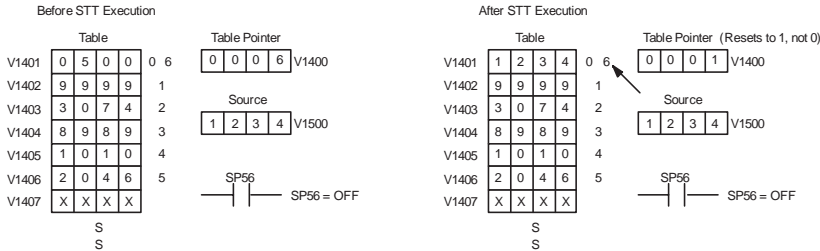


S  
S  
S

## Scan N+5



## Scan N+6



## Remove from Table (RFT)

The Remove From Table instruction pops a value off of a table and stores it in a V memory location. When a value is removed from the table all other values are shifted up 1 location. The first V memory location in the table contains the table length counter. The table counter decrements by 1 each time the instruction is executed. If the length counter is zero or greater than the maximum table length (specified in the first level of the accumulator stack) the instruction will not execute and SP56 will be on.



The instruction will be executed once per scan provided the input remains on. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Remove From Table function.

- Step 1: Load the length of the table (number of V memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.
- Step 2: Load the starting V memory location for the table into the accumulator. (Remember, the starting location of the table is used as the table length counter.) This parameter must be a HEX value.
- Step 3: Insert the RFT instructions which specifies destination V memory location (Vaaa). This is where the value will be moved to.

**Helpful Hint:** — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

**Helpful Hint:**— The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

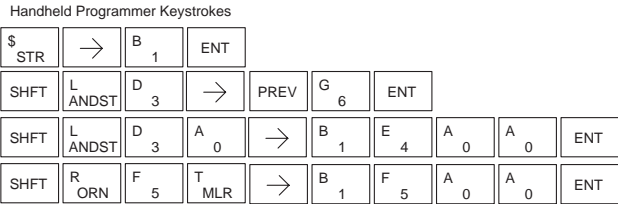
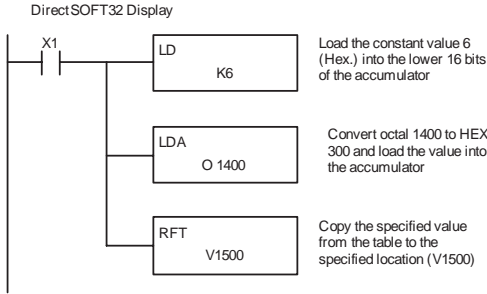
**Helpful Hint:** — The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the table counter should be between 1 and 6. If the value is outside of this range or zero, the data will not be moved from the table. Also, a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

Operand Data Type	DL06 Range
V memory ..... V	<b>aaa</b> See memory map
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.



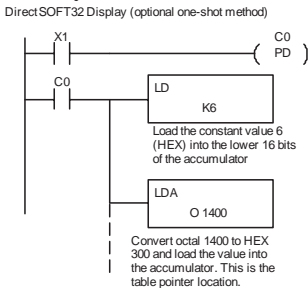
Since the table counter specifies the range of data that will be removed from the table, it is important to understand how the table locations are numbered. If you examine the example table, you'll notice that the data locations are numbered from the top of the table. For example, if the table counter started at 6, then all six of the locations would be affected during the instruction execution.

Table					
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Counter					
0	0	0	6	V1400	

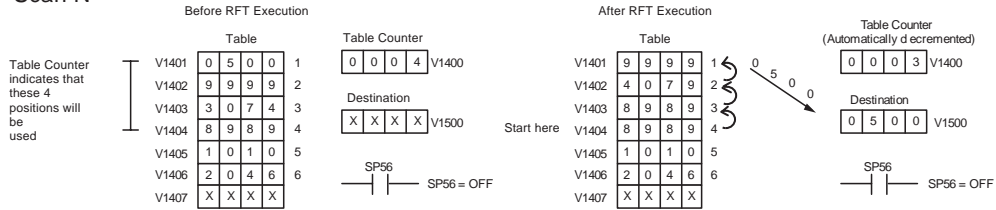
Destination					
X	X	X	X	V1500	

Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the pointer decrements automatically, the data would be removed from the table very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to remove one value each time the input contact transitions from low to high.

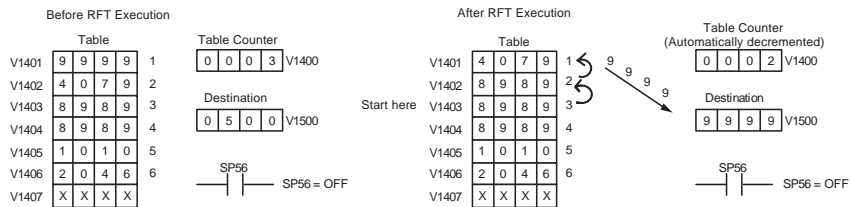


The following diagram shows the scan-by-scan results of the execution for our example program. In our example we're showing the table counter set to 4 initially. (Remember, you can set the table counter to any value that is within the range of the table.) The table counter automatically decrements from 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.

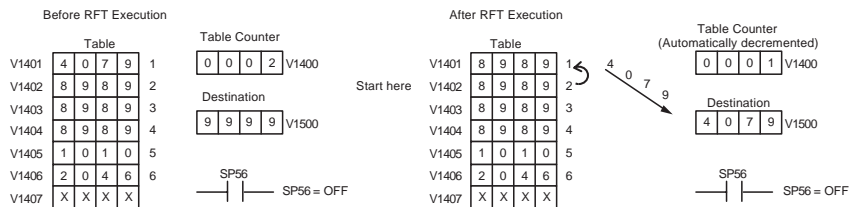
## Scan N



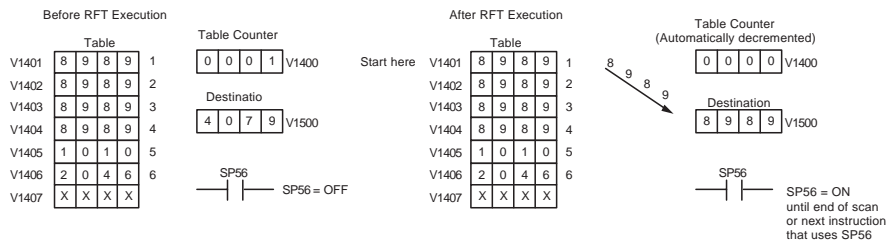
## Scan N+1



## Scan N+2



## Scan N+3



### Add to Top (ATT)

The Add To Top instruction pushes a value on to a V memory table from a V memory location. When the value is added to the table all other values are pushed down 1 location.



The instruction will be executed once per scan provided the input remains on. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Add To Top function.

- Step 1: Load the length of the table (number of V memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.
- Step 2 Load the starting V memory location for the table into the accumulator. (Remember, the starting location of the table is used as the table length counter.) This parameter must be a HEX value.
- Step 3: Insert the ATT instructions which specifies source V memory location (Vaaa). This is where the value will be moved from.

Helpful Hint:— The instruction will be executed every scan if the input logic is on. If you do not want the instruction to execute for more than one scan, a one shot (PD) should be used in the input logic.

Helpful Hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Helpful Hint: — The table counter value should be set to indicate the starting point for the operation. Also, it must be set to a value that is within the length of the table. For example, if the table is 6 words long, then the allowable range of values that could be in the table counter should be between 1 and 6. If the value is outside of this range or zero, the data will not be moved into the table. Also, a one shot (PD) should be used so the value will only be set in one scan and will not affect the instruction operation.

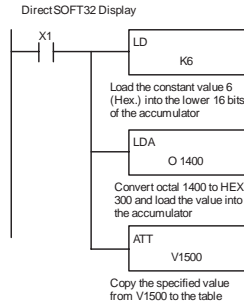
Operand Data Type	DL06 Range
	<b>aaa</b>
V memory ..... V	See memory map

Discrete Bit Flags	Description
SP56	On when the table counter is equal to the table size.

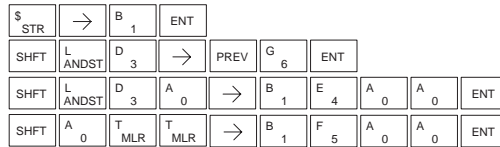


**NOTE:** Status flags (SPs) are only valid until: — another instruction that uses the same flag is executed, or — the end of the scan The pointer for this instruction can be set to start anywhere in the table. It is not set automatically. You have to load a value into the pointer somewhere in your program.

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



For the ATT instruction, the table counter determines the number of additions that can be made before the instruction will stop executing. So, it is helpful to understand how the system uses this counter to control the execution.

For example, if the table counter was set to 2, and the table length was 6 words, then there could only be 4 additions of data before the execution was stopped. This can easily be calculated by:

**Table length – table counter = number of executions**

Also, our example uses a normal input contact (X1) to control the execution. Since the CPU scan is extremely fast, and the table counter increments automatically, the data would be moved into the table very quickly. If this is a problem for your application, you have an option of using a one-shot (PD) to add one value each time the input contact transitions from low to high.

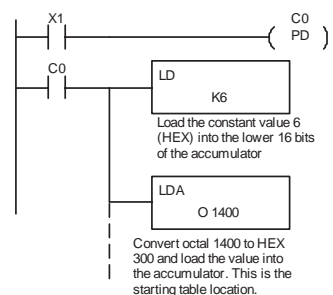
Table					
V1401	0	5	0	0	1
V1402	9	9	9	9	2
V1403	3	0	7	4	3
V1404	8	9	8	9	4
V1405	1	0	1	0	5
V1406	2	0	4	6	6
V1407	X	X	X	X	

Table Counter				
0	0	0	2	V1400

Data Source				
X	X	X	X	V1500

(e.g.: 6 - 2 = 4)

DirectSOFT32 Display (optional one-shot method)

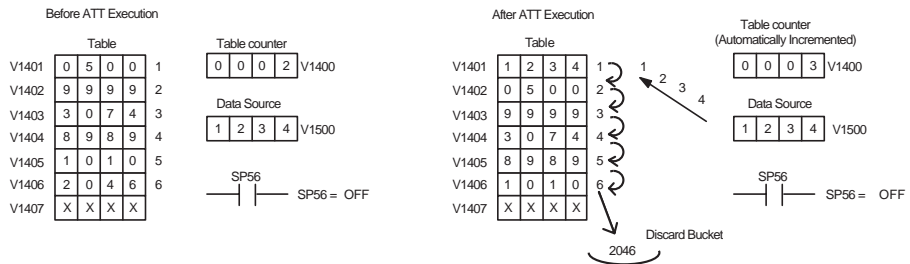


## Chapter 5: Standard RLL Instructions - Table Instructions

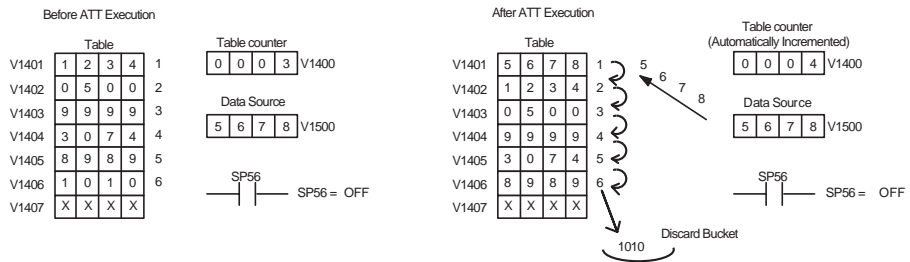
The following diagram shows the scan-by-scan results of the execution for our example program. The table counter is set to 2 initially, and it will automatically increment from 2 – 6 as the instruction is executed. Notice how SP56 comes on when the table counter is 6, which is equal to the table length. 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

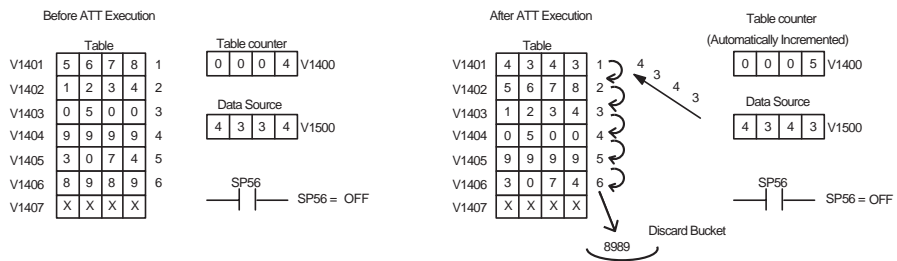
#### Scan N



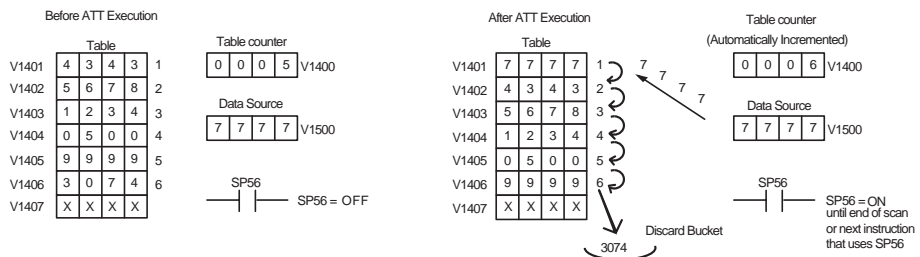
#### Scan N+1



#### Scan N+2



#### Scan N+3



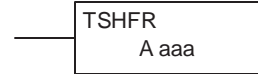
## Table Shift Left (TSHFL)

The Table Shift Left instruction shifts all the bits in a V-memory table to the left, the specified number of bit positions.

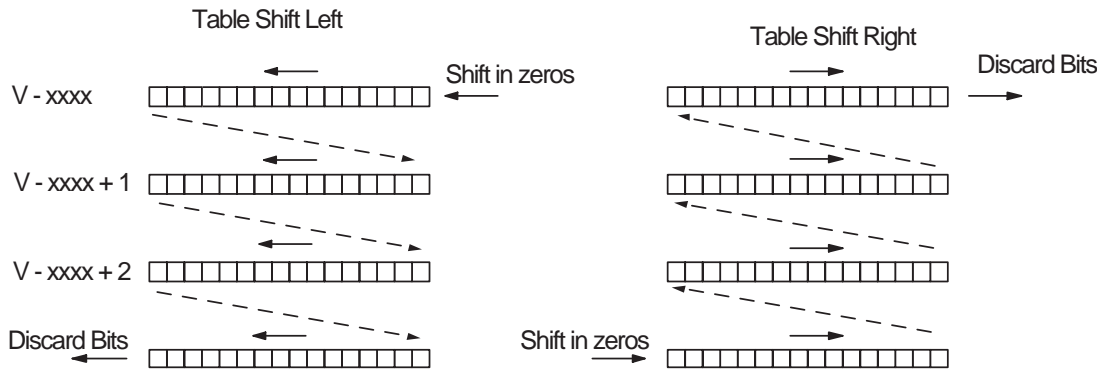


## Table Shift Right (TSHFR)

The Table Shift Right instruction shifts all the bits in a V-memory table to the right, a specified number of bit positions.



The following description applies to both the Table Shift Left and Table Shift Right instructions. A table is just a range of V-memory locations. The Table Shift Left and Table Shift Right instructions shift bits serially throughout the entire table. Bits are shifted out the end of one word and into the opposite end of an adjacent word. At the ends of the table, bits are either discarded, or zeros are shifted into the table. The example tables below are arbitrarily four words long.



- Step 1: Load the length of the table (number of V memory locations) into the first level of the accumulator stack. This parameter must be a HEX value, 0 to FF.
- Step 2: Load the starting V memory location for the table into the accumulator. This parameter must be a HEX value. You can use the LDA instruction to convert an octal address to hex.
- Step 3: Insert the Table Shift Left or Table shift Right instruction. This specifies the number of bit positions you wish to shift the entire table. The number of bit positions must be in octal.

Helpful hint: — Remember that each V memory location contains 16 bits. So, the bits of the first word of the table are numbered from 0 to 17 octal. If you want to shift the entire table by 20 bits, that is 24 octal. SP 53 will be set if the number of bits to be shifted is larger than the total bits contained within the table. Flag 67 will be set if the last bit shifted (just before it is discarded) is a "1".

Operand Data Type	DL06 Range
	<b>aaa</b>
V memory .....	See memory map



Discrete Bit Flags	Description
SP53	On when the number of bits to be shifted is larger than the total bits contained within the table
SP67	On when the last bit shifted (just before it is discarded) is a "1."



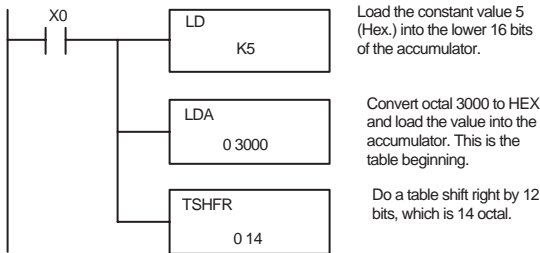
**NOTE:** Status flags are only valid until: — the end of the scan  
— another instruction that uses the same flag is executed.

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.

V 3000	V 3000
1 2 3 4	6 7 8 1
5 6 7 8	1 2 2 5
1 1 2 2	3 4 4 1
3 3 4 4	5 6 6 3
5 5 6 6	0 0 0 5

DirectSOFT 32



Handheld Programmer Keystrokes

\$ STR	→	A <sub>0</sub>	ENT								
SHFT	L ANDST	D <sub>3</sub>	→	PREV	F <sub>5</sub>	ENT					
SHFT	L ANDST	D <sub>3</sub>	A <sub>0</sub>	→	D <sub>3</sub>	A <sub>0</sub>	A <sub>0</sub>	A <sub>0</sub>	ENT		
SHFT	T MLR	SHFT	S RST	H <sub>7</sub>	F <sub>5</sub>	R ORN	→	NEXT	B <sub>1</sub>	E <sub>4</sub>	ENT

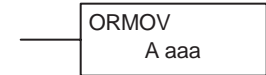
## AND Move (ANDMOV)

The AND Move instruction copies data from a table to the specified memory location, ANDing each word with the accumulator data as it is written.



## OR Move (ORMOV)

The Or Move instruction copies data from a table to the specified memory location, ORing each word with the accumulator contents as it is written.



## Exclusive OR Move (XORMOV)

The Exclusive OR Move instruction copies data from a table to the specified memory location, XORing each word with the accumulator value as it is written.



The following description applies to the AND Move, OR Move, and Exclusive OR Move instructions. A table is just a range of V-memory locations. These instructions copy the data of a table to another specified location, performing 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	DL06 Range
	<b>aaa</b>
V memory ..... V	See memory map

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.

DirectSOFT 32

## Handheld Programmer Keystrokes

\$	STR	→	A <sub>0</sub>	ENT						
SHFT	L ANDST	D <sub>3</sub>	→	PREV	C <sub>2</sub>	ENT				
SHFT	L ANDST	D <sub>3</sub>	A <sub>0</sub>	→	D <sub>3</sub>	A <sub>0</sub>	A <sub>0</sub>	A <sub>0</sub>	ENT	
SHFT	L ANDST	D <sub>3</sub>	→	PREV	G <sub>6</sub>	G <sub>6</sub>	G <sub>6</sub>	G <sub>6</sub>	ENT	
V AND	SHFT	M ORST	O INST#	V AND	→	D <sub>3</sub>	B <sub>1</sub>	A <sub>0</sub>	A <sub>0</sub>	ENT

The example to the right shows a table of two words at V3000 and logically ORs it with K8888. The copy of the table at V3100 shows the result of the OR operation for each word.

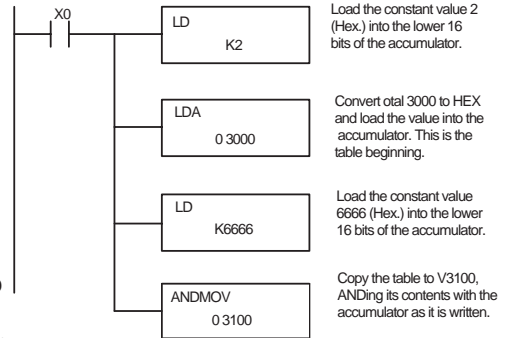
The program to the right performs the ORMOV example above. It assumes that the data in the table at V3000 – V3001 already exists. First we load the table length (two words) into the accumulator. Next we load the starting address of the source table, using the LDA instruction. Then we load the data into the accumulator to be ORed with the table. In the ORMOV command, we specify the table destination, V3100.

## Handheld Programmer Keystrokes

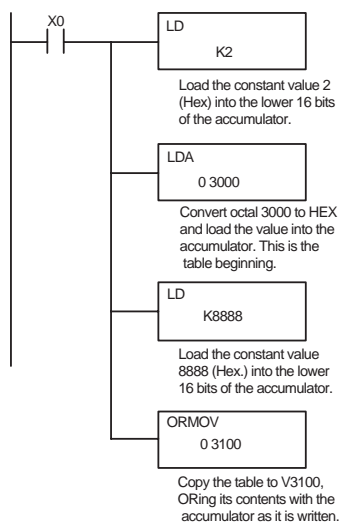
[illegible]

The example to the right shows a table of two words at V3000 and logically XORs it with K3333. The copy of the table at V3100 shows the result of the XOR operation for each word.

The ladder program example for the XORMOV is similar to the one above for the ORMV. Just use the XORMOV instruction. On the handheld programmer, you must use the SHIFT key and spell "XORMOV" explicitly.



DirectSOFT 32



## Find Block (FINDB)

The Find Block instruction searches for an occurrence of a specified block of values in a V memory table. The function parameters are loaded into the first and second levels of the accumulator stack and the accumulator by three additional instructions. If the block is found, its starting address will be stored in the accumulator. If the block is not found, flag SP53 will be set.



Operand Data Type	DL06 Range
	<b>aaa</b>
V memory . . . . . V	See memory map
V memory . . . . . P	See memory map

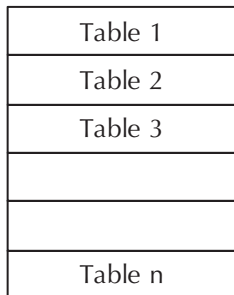
  

Discrete Bit Flags	Description
SP53	On when the Find Block instruction was executed but did not find the block of data in table specified

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.

Start Addr.



Number  
of words

Start Addr.

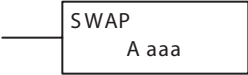


Number  
of bytes

End Addr.

Swap (SWAP)

The Swap instruction exchanges the data in two tables of equal length.



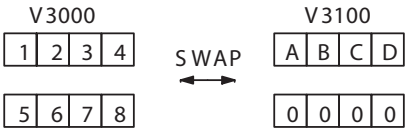
The following description applies to both the Set Bit and Reset Bit table instructions.

- 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.

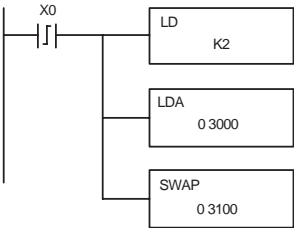
Operand Data Type	DL06 Range
	<b>aaa</b>
V memory ..... V	See memory map

The example to the right shows a table of two words at V3000. We will swap its contents with another table of two words at 3100 by using the Swap instruction. The required ladder program is given below.



The example program below uses a PD contact (triggers for one scan for off-to-on transition). First, we load the length of the tables (two words) into the accumulator. Then we load the address of the first table (V3000) into the accumulator using the LDA instruction, converting the octal address to hex. Note that it does not matter which table we declare “first”, because the swap results will be the same.

DirectSOF 32



Load the constant value 2 (Hex.) into the lower 16 bits of the accumulator.

Convert octal 3000 to HEX and load the value into the accumulator. This is the table beginning.

Swap the contents of the table in the previous instruction with the one at V3100.

Handheld Programmer Keystrokes

\$	STR	SHFT	P	CV	D	3	→	A	0	ENT
SHFT	L	ANDST	D	3	→	PREV	C	2	ENT	
SHFT	L	ANDST	D	3	→	A	0	A	0	ENT
SHFT	S	RST	SHFT	W	ANDN	A	0	P	CV	→
						D	3	B	1	A
								A	0	ENT

## Clock / Calendar Instructions

### Date (DATE)

The Date instruction can be used to set the date in the CPU. The instruction requires two consecutive V memory locations (Vaaa) to set the date. If the values in the specified locations are not valid, the date will not be set. The current date can be read from 4 consecutive V memory locations (V7771-V7774).

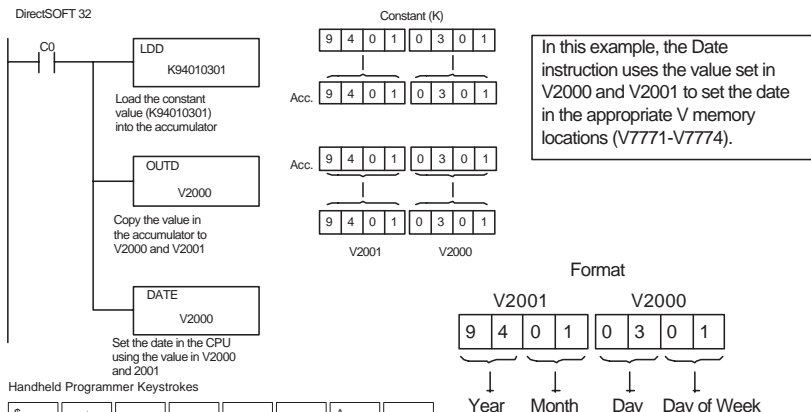


In the following example, when C0 is on, the constant value (K94010301) is loaded into the accumulator using the Load Double instruction (C0 should be a contact from a one shot (PD) instruction). The value in the accumulator is output to V2000 using the Out Double instruction. The Date instruction uses the value in V2000 to set the date in the CPU.

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	DL06 Range
<b>A</b>	<b>aaa</b>
V memory..... V	See memory map



Handheld Programmer Keystrokes

\$ STR	→	NEXT	NEXT	NEXT	NEXT	A <sub>0</sub>	ENT	Year				Month	
SHFT	L ANDST	D <sub>3</sub>	D <sub>3</sub>	→	PREV	J <sub>9</sub>	E <sub>4</sub>	A <sub>0</sub>	B <sub>1</sub>	ENT			
A <sub>0</sub>	D <sub>3</sub>	A <sub>0</sub>	B <sub>1</sub>	ENT									
GX OUT	SHFT	D <sub>3</sub>	→	C <sub>2</sub>	A <sub>0</sub>	A <sub>0</sub>	A <sub>0</sub>	ENT					
SHFT	D <sub>3</sub>	A <sub>0</sub>	T MLR	E <sub>4</sub>	→	C <sub>2</sub>	A <sub>0</sub>	A <sub>0</sub>	A <sub>0</sub>	A <sub>0</sub>	ENT		

Time (TIME)

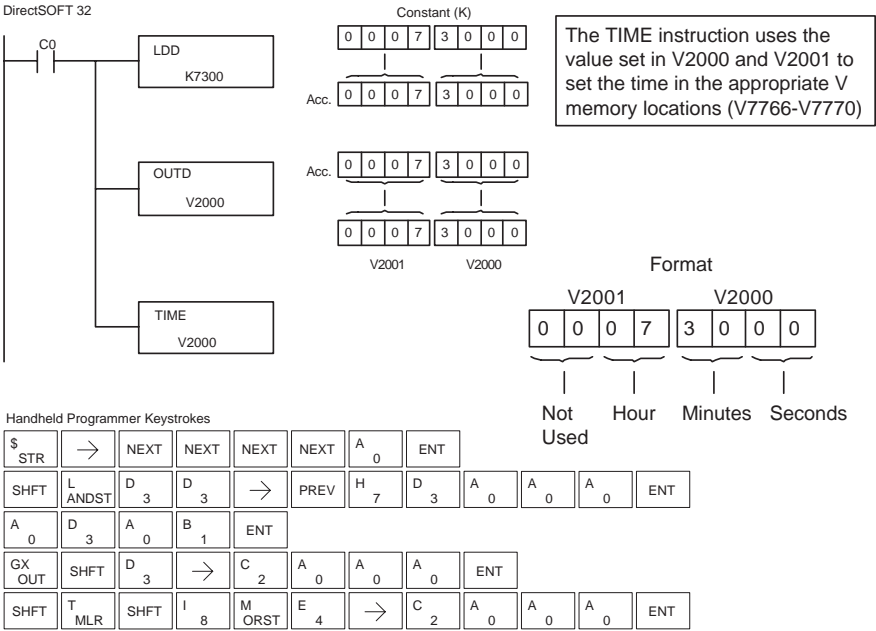
The Time instruction can be used to set the time (24 hour clock) in the CPU. The instruction requires two consecutive V memory locations (Vaaa) which are used to set the time. If the values in the specified locations are not valid, the time will not be set. The current time can be read from memory locations V7747 and V7766-V7770.



Date	Range	VMemory 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	DL06 Range
<b>A</b>	<b>aaa</b>
V memory .....	See memory map

In the following example, when C0 is on, the constant value (K73000) is loaded into the accumulator using the Load Double instruction (C0 should be a contact from a one shot (PD) instruction). The value in the accumulator is output to V2000 using the Out Double instruction. The Time instruction uses the value in V2000 to set the time in the CPU.



# CPU Control Instructions

## No Operation (NOP)

The No Operation is an empty (not programmed) memory location.

—( NOP )

Direct SOFT32



Handheld Programmer Keystrokes



## End (END)

The End instruction marks the termination point of the normal program scan. An End instruction is required at the end of the main program body. If the End instruction is omitted an error will occur and the CPU will not enter the Run Mode. Data labels, subroutines and interrupt routines are placed after the End instruction. The End instruction is not conditional; therefore, no input contact is allowed.

—( END )

Direct SOFT32



Handheld Programmer Keystrokes



## Stop (STOP)

The Stop instruction changes the operational mode of the CPU from Run to Program (Stop) mode. This instruction is typically used to stop PLC operation in an error condition.

—( STOP )

In the following example, when C0 turns on, the CPU will stop operation and switch to the program mode.

DirectSOFT32



Handheld Programmer Keystrokes



Discrete Bit Flags	Description
SP16	On when the DL06 goes into the TERM_PRG mode.
SP53	On when the DL06 goes into the PRG mode.



Reset Watch Dog Timer (RSTWT)

The Reset Watch Dog Timer instruction resets the CPU scan timer. The default setting for the watch dog timer is 200ms. Scan times very seldom exceed 200ms, but it is possible. For/next loops, subroutines, interrupt routines, and table instructions can be programmed such that the scan becomes longer than 200ms. When instructions are used in a manner that could exceed the watch dog timer setting, this instruction can be used to reset the timer.

—(RSTWT)

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 32



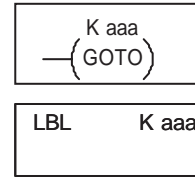
Handheld Programmer Keystrokes

SHIFT	R ORN	S RST	T MLR	W ANDN	T MLR	ENT
-------	----------	----------	----------	-----------	----------	-----

## Program Control Instructions

### Goto Label (GOTO) (LBL)

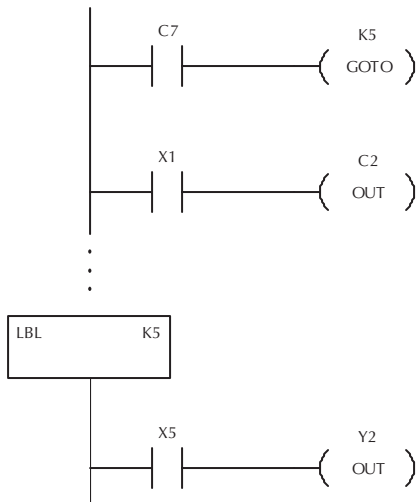
The Goto / Label skips all instructions between the Goto and the corresponding LBL instruction. The operand value for the Goto and the corresponding LBL instruction are the same. The logic between Goto and LBL instruction is not executed when the Goto instruction is enabled. Up to 256 Goto instructions and 256 LBL instructions can be used in the program.



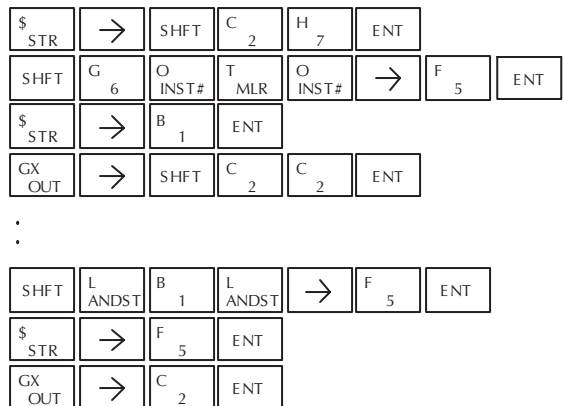
Operand Data Type	DL06 Range
	<b>aaa</b>
Constant.....K	1-FFFF

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.

DirectSOFT32



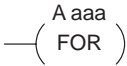
Handheld Programmer Keystrokes



For / Next (FOR) (NEXT)

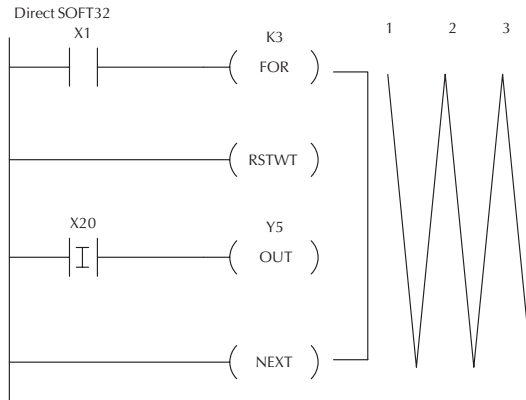
The For and Next instructions are used to execute a section of ladder logic between the For and Next instruction a specified numbers of times. When the For instruction is enabled, the program will loop the specified number of times. If the For instruction is not energized the section of ladder logic between the For and Next instructions is not executed.

For / Next instructions cannot be nested. The normal I/O update and CPU housekeeping 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	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Constant ..... K	1-9999

In the following example, when X1 is on, the application program inside the For / Next loop will be executed three times. If X1 is off the program inside the loop will not be executed. The immediate instructions may or may not be necessary depending on your application. Also, The RSTWT instruction is not necessary if the For / Next loop does not extend the scan time larger the Watch Dog Timer setting. For more information on the Watch Dog Timer, refer to the RSTWT instruction.

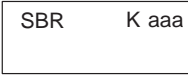
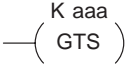


Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT			
SHFT	F 5	O INST#	R ORN	→	D 3	ENT
SHFT	R ORN	S RST	T MLR	W ANDN	T MLR	ENT
\$ STR	SHFT	I 8	→	C 2	A 0	ENT
GX OUT	→	F 5	ENT			
SHFT	N TMR	E 4	X SET	T MLR	ENT	

Goto Subroutine (GTS) (SBR)

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 256 GTS instructions and 256 SBR instructions used in a program. The GTS instructions can be nested up to 8 levels. An error E412 will occur if the maximum limits are exceeded. Typically this will be used in an application where a block of program logic may be slow to execute and is not required to execute every scan. The subroutine label and all associated logic is placed after the End statement in the program. When the subroutine is called from the main program, the CPU will execute the subroutine (SBR) with the same constant number (K) as the GTS instruction 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.

Subroutine Return (RT)

Operand Data Type	DL06 Range
	aaa
Constant ..... K	1-FFFF

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).

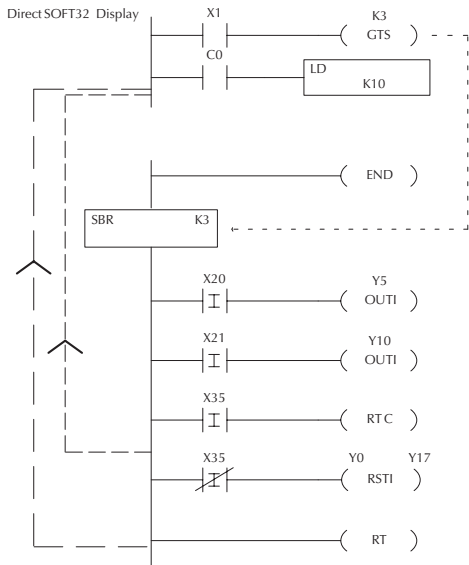


Subroutine Return Conditional (RTC)

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.

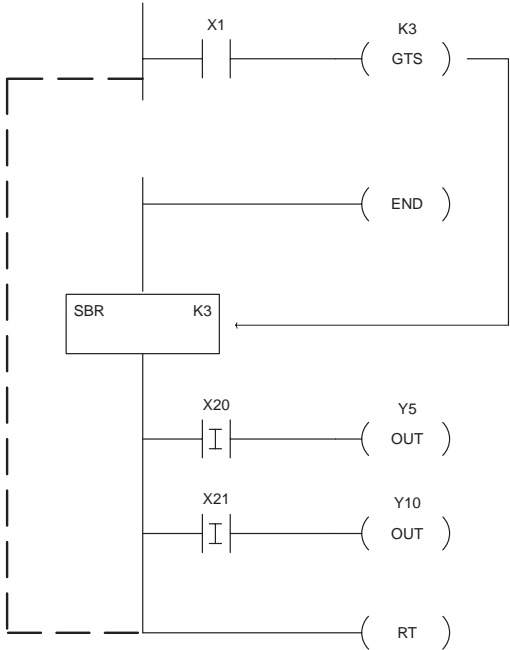


Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT						
SHFT	G 6	T MLR	S RST	→	D 3	ENT			
SHFT	E 4	N TMR	D 3	ENT					
SHFT	S RST	SHFT	B 1	R ORN	→	D 3	ENT		
\$ STR	SHFT	I 8	→	C 2	A 0	ENT			
GX OUT	SHFT	I 8	→	F 5	ENT				
\$ STR	SHFT	I 8	→	C 2	B 1	ENT			
GX OUT	SHFT	I 8	→	B 1	A 0	ENT			
\$ STR	SHFT	I 8	→	D 3	F 5	ENT			
SHFT	R ORN	T MLR	C 2	ENT					
SP STRN	SHFT	I 8	→	D 3	F 5	ENT			
S RST	SHFT	I 8	→	A 0	→	B 1	H 7	ENT	
SHFT	R ORN	T MLR	ENT						

In the following example, when X1 is on, Subroutine K3 will be called. The CPU will jump to the Subroutine Label K3 and the ladder logic in the subroutine will be executed. The CPU will return to the main body of the program after the RT instruction is executed.

Direct SOFT32



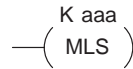
Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT				
SHFT	G 6	T MLR	S RST	→	D 3	ENT	

SHFT	E 4	N TMR	D 3	ENT				
SHFT	S RST	SHFT	B 1	R ORN	→	D 3	ENT	
\$ STR	SHFT	I 8	→	C 2	A 0	ENT		
GX OUT	→	F 5	ENT					
\$ STR	SHFT	I 8	→	C 2	B 1	ENT		
GX OUT	→	B 1	A 0	ENT				
SHFT	R ORN	T MLR	ENT					

## Master Line Set (MLS)

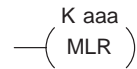
The Master Line Set instruction allows the program to control sections of ladder logic by forming a new power rail controlled by the main left power rail. The main left rail is always master line 0. When a MLS K1 instruction is used, a new power rail is created at level 1. Master Line Sets and Master Line Resets can be used to nest power rails up to seven levels deep.



Operand Data Type	DL06 Range
	<b>aaa</b>
Constant .....K	1-7

## Master Line Reset (MLR)

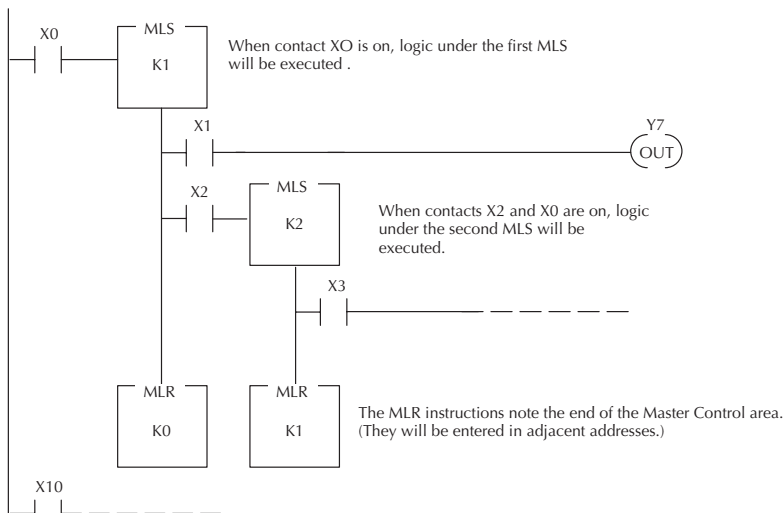
The Master Line Reset instruction marks the end of control for the corresponding MLS instruction. The MLR reference is one less than the corresponding MLS.



Operand Data Type	DL06 Range
	<b>aaa</b>
Constant .....K	0-7

## Understanding Master Control Relays

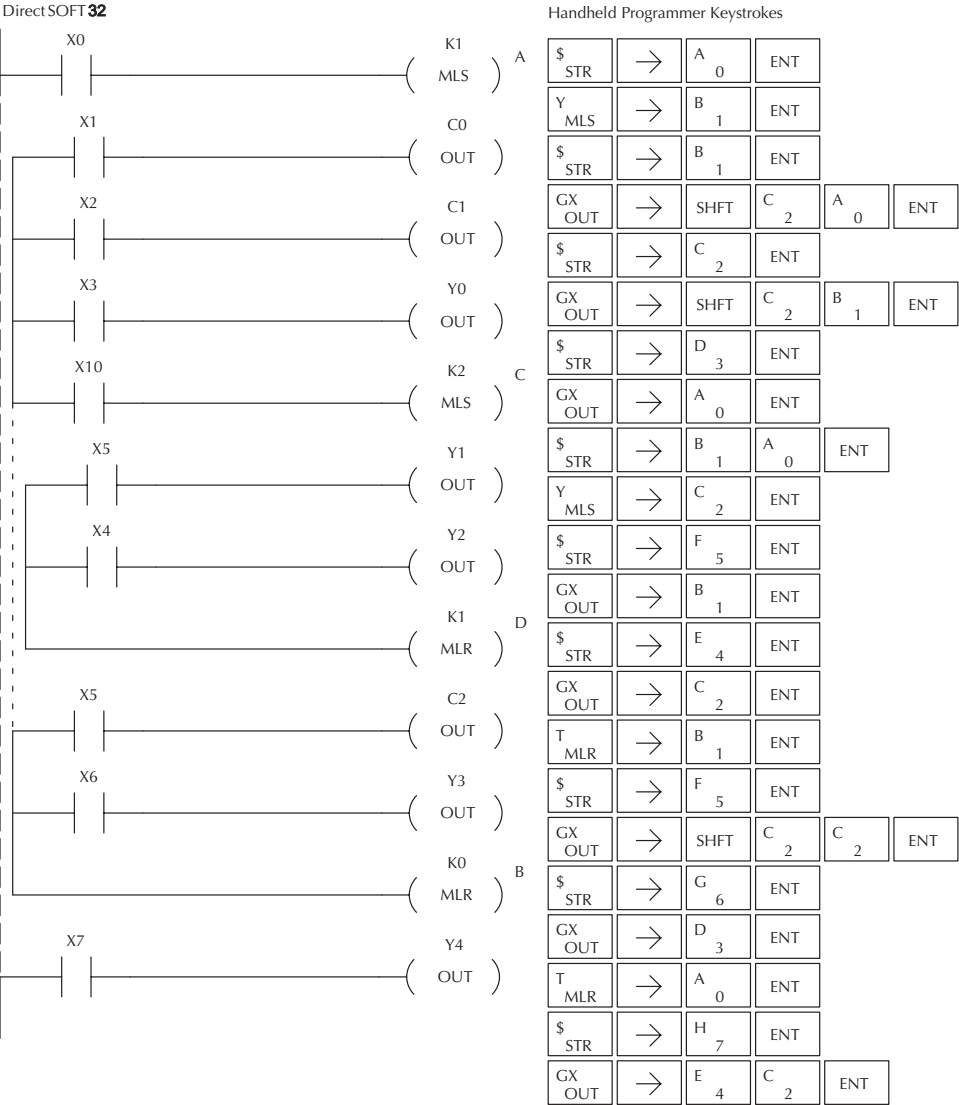
The Master Line Set (MLS) and Master Line Reset (MLR) instructions allow you to quickly enable (or disable) sections of the RLL program. This provides program control flexibility. The following example shows how the MLS and MLR instructions operate by creating a sub power rail for control logic.





MLS/MLR Example

In the following MLS/MLR example logic between the first MLS K1 (A) and MLR K0 (B) will function only if input X0 is on. The logic between the MLS K2 (C) and MLR K1 (D) will function only if input X10 and X0 is on. The last rung is not controlled by either of the MLS coils.



## Interrupt Instructions

### Interrupt (INT)

The Interrupt instruction allows a section of ladder logic to be placed below the main body of the program and executed only when needed. High-Speed I/O Modes 10, 20, and 40 can generate an interrupt. With Mode 40, you may select an external interrupt (input X0), or a time-based interrupt (3–999 ms).

INT	O aaa
-----	-------

Typically, interrupts are used in an application when a fast response to an input is needed or a program section must 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 an interrupt occurs, the CPU will complete execution of the current instruction it is processing in ladder logic, then execute the interrupt routine. After interrupt routine execution, the ladder program resumes from the point at which it was interrupted.

See Chapter 3, the section on Mode 40 (Interrupt) Operation for more details on interrupt configuration. In the DL06, only one software interrupt is available. The software interrupt uses interrupt #00 (INT 0), which means the hardware interrupt #0 and the software interrupt cannot be used together. Hardware interrupts are labeled in octal to correspond with the hardware input signal (e.g. X1 will initiate INT 1).

Operand Data Type	DL06 Range
	aaa
Constant ..... 0	0-3

### Interrupt Return (IRT)

An Interrupt Return is normally executed as the last instruction in the interrupt routine. It returns the CPU to the point in the main program from which it was called. The Interrupt Return is a stand-alone instruction (no input contact on the rung).

—( IRT )

### Interrupt Return Conditional (IRTC)

The Interrupt Return Conditional instruction is a optional instruction used with an input contact to implement a conditional return from the interrupt routine. The Interrupt Return is required to terminate the interrupt routine.

—( IRTC )

### Enable Interrupts (ENI)

The Enable Interrupt instruction is placed in the main ladder program (before the End instruction), enabling the interrupt. The interrupt remains enabled until the program executes a Disable Interrupt instruction.

—( ENI )

Disable Interrupts (DISI)

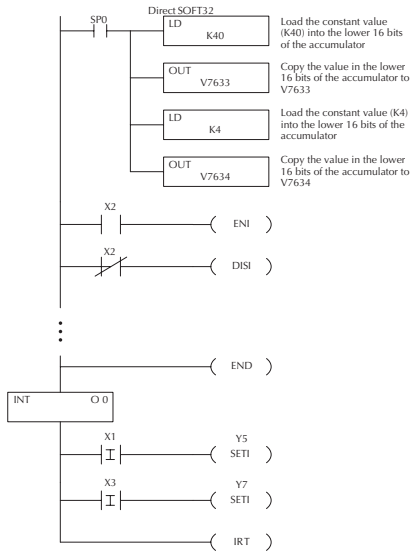
A Disable Interrupt instruction in the main body of the application program (before the End instruction) will disable the interrupt (either external or timed). The interrupt remains disabled until the program executes an Enable Interrupt instruction.

( DISI )

External Interrupt Program Example

In the following example, we do some initialization on the first scan, using the first-scan contact SP0. The interrupt feature is the HSIO Mode 40. Then we configure X0 as the external interrupt by writing to its configuration register, V7634. See Chapter 3, Mode 40 Operation for more details.

During program execution, when X2 is on the interrupt is enabled. When X2 is off the interrupt will be disabled. When an interrupt signal (X0) occurs the CPU will jump to the interrupt label INT O 0. The application ladder logic in the interrupt routine will be performed. The CPU will return to the main body of the program after the IRT instruction is executed.



Handheld Programmer Keystrokes

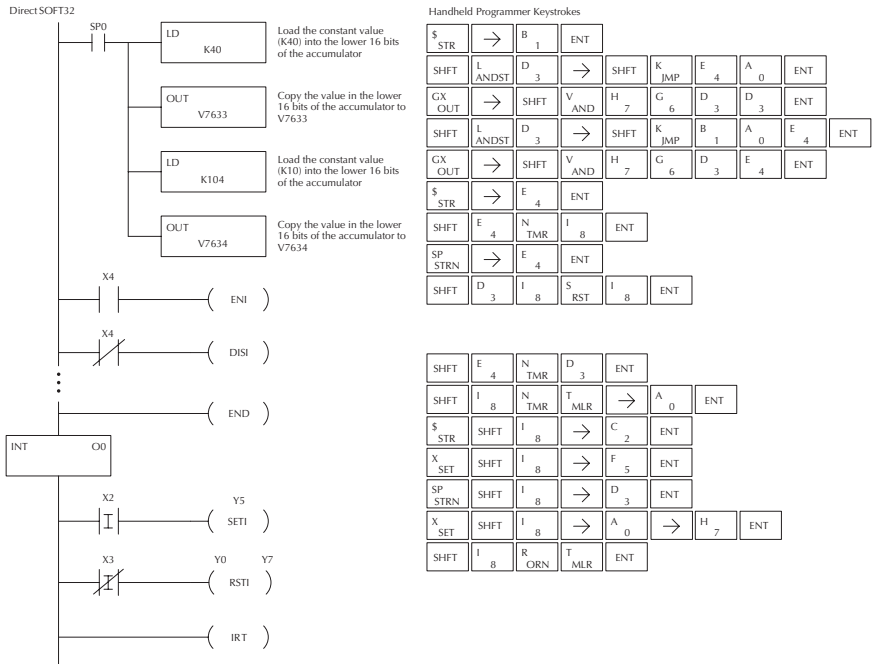
\$ STR	→	SHIFT	SP STRN	A 0	ENT
SHIFT	L ANDST	D 3	→	SHIFT	K JMP
GX OUT	→	SHIFT	V AND	H 7	G 6
SHIFT	L ANDST	D 3	→	SHIFT	K JMP
GX OUT	→	SHIFT	V AND	H 7	G 6
\$ STR	→	C 2	ENT	D 3	E 4
SHIFT	E 4	N TMR	I 8	ENT	
SP STRN	→	C 2	ENT		
SHIFT	D 3	I 8	S RST	I 8	ENT

SHIFT	E 4	N TMR	D 3	ENT	
SHIFT	I 8	N TMR	T MLR	→	A 0
\$ STR	SHIFT	I 8	→	B 1	ENT
X SET	SHIFT	I 8	→	F 5	ENT
\$ STR	SHIFT	I 8	→	D 3	ENT
X SET	SHIFT	I 8	→	H 7	ENT
SHIFT	I 8	R ORN	T MLR	ENT	

## Timed Interrupt Program Example

In the following example, we do some initialization on the first scan, using the first-scan contact SP0. The interrupt feature is the HSIO Mode 40. Then we configure the HSIO timer as a 10 mS interrupt by writing K104 to the configuration register for X0 (V7634). See Chapter 3, Mode 40 Operation for more details.

When X4 turns on, the interrupt will be enabled. When X4 turns off, the interrupt will be disabled. Every 10 mS the CPU will jump to the interrupt label INT 00. The application ladder logic in the interrupt routine will be performed. If X3 is not on Y0–Y7 will be reset to off and then the CPU will return to the main body of the program.



## Independent Timed Interrupt

Interrupt 00 is also available as an interrupt. This interrupt is independent of the HSIO features. Interrupt 00 uses an internal timer that is configured in V memory location V7647. The interrupt period can be adjusted from 5 to 9999 mS. Once the interrupt period is set and the interrupt is enabled in the program, the CPU will continuously call the interrupt routine based on the time setting in V7647.

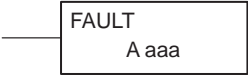
Note: Independent timed interrupt must be Int. 0.

Input	Configuration Register	Function	Hex Code Required
—	V7647	High-Speed Timed Interrupt	xxxx (xxxx = timer setting) 5 - 9999 mS (BCD)

# Message Instructions

## Fault (FAULT)

The Fault instruction is used to display a message on the handheld programmer, the optional LCD display or in the *DirectSOFT* status bar. The message has a maximum of 23 characters and can be either V memory data, numerical constant data or ASCII text.



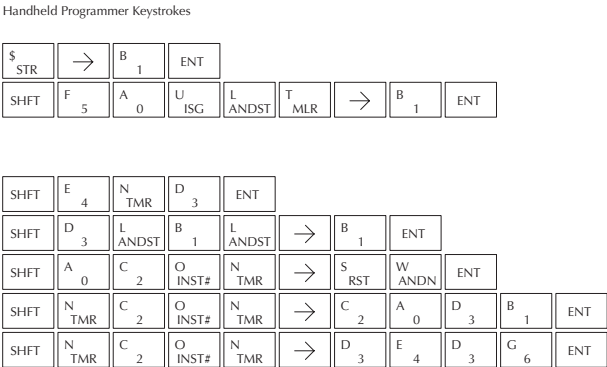
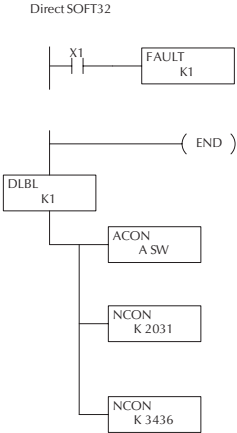
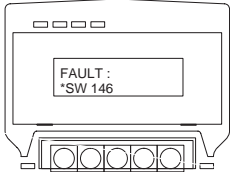
To display the value in a V memory location, specify the V memory location in the instruction. To display the data in ACON (ASCII constant) or NCON (Numerical constant) instructions, specify the constant (K) value for the corresponding data label area.

Operand Data Type	DL06 Range
..... A	aaa
V memory ..... V	See memory map
Constant ..... K	1-FFFF

Discrete Bit Flags	Description
SP50	On when the FAULT instruction is executed

## Fault Example

In the following example when X1 is on, the message SW 146 will display on the handheld programmer. The NCONs use the HEX ASCII equivalent of the text to be displayed. (The HEX ASCII for a blank is 20, a 1 is 31, 4 is 34 ...)



## Data Label (DLBL)

The Data Label instruction marks the beginning of an ASCII / numeric data area. DLBLs are programmed after the End statement. A maximum of 64 DLBL instructions can be used in a program. Multiple NCONs and ACONs can be used in a DLBL area.

DLBL      K aaa

Operand Data Type	DL06 Range
	<b>aaa</b>
Constant ..... K	1-FFFF

## ASCII Constant (ACON)

The ASCII Constant instruction is used with the DLBL instruction to store ASCII text for use with other instructions. Two ASCII characters can be stored in an ACON instruction. If only one character is stored in a ACON a leading space will be inserted.

ACON  
A aaa

Operand Data Type	DL06 Range
	<b>aaa</b>
ASCII ..... A	0-9 A-Z

## Numerical Constant (NCON)

The Numerical Constant instruction is used with the DLBL instruction to store the HEX ASCII equivalent of numerical data for use with other instructions. Two digits can be stored in an NCON instruction.

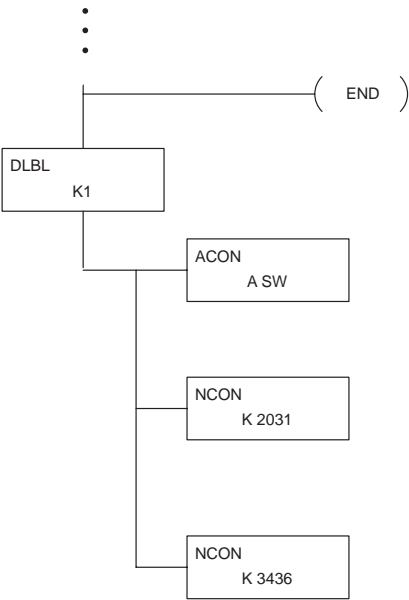
NCON  
K aaa

Operand Data Type	DL06 Range
	<b>aaa</b>
Constant ..... K	0-FFFF

Data Label Example

In the following example, an ACON and two NCON instructions are used within a DLBL instruction to build a text message. See the FAULT instruction for information on displaying messages. The DV-1000 Manual also has information on displaying messages.

Direct SOFT32



Handheld Programmer Keystrokes

SHFT	E 4	N TMR	D 3	ENT													
SHFT	D 3	L ANDST	B 1	L ANDST	→	B 1	ENT										
SHFT	A 0	C 2	O INST#	N TMR	→	S RST	W ANDN	ENT									
SHFT	N TMR	C 2	O INST#	N TMR	→	C 2	A 0	D 3	B 1	ENT							
SHFT	N TMR	C 2	O INST#	N TMR	→	D 3	E 4	D 3	G 6	ENT							

## Print Message (PRINT)

The Print Message instruction prints the embedded text or text/data variable message to the specified communications port (Port 2 on the DL06 CPU), which must have the communications port configured.

```
PRINT    A aaa
"Hello, this is a PLC message"
```

Operand Data Type	DL06 Range
<b>A</b>	<b>aaa</b>
Constant .....	2

You may recall from the CPU specifications in Chapter 3 that the DL06's ports are capable of several protocols. Port 1 cannot be configured for the non-sequence protocol. To configure port 2 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 *DirectSOFT32*, choose the PLC menu, then Setup, then Setup Secondary Comm Port.

- **Port:** From the port number list box at the top, choose "Port 2".
- **Protocol:** Click the check box to the left of "Non-sequence", and then you'll see the dialog box shown below.

- **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 *DirectSOFT32* to use to store the port setup information. You will need to reserve 9 words in V-memory for this purpose. Select "Always use for printing" if it applies.



Then click the button indicated to send the Port 2 configuration to the CPU, and click Close. Then see Chapter 3 for port wiring information, in order to connect your printer to the DL06.



Port 2 on the DL06 has standard RS232 levels, and should work with most printer serial input connections.

**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

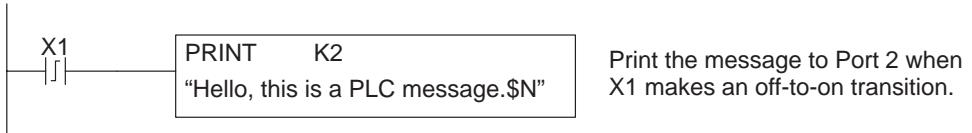
" \$ R \$ L " Length 2 with one CR and one LF

" \$ 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. Error code 499 will occur in the CPU when the print instruction contains invalid text or no quotations. It is important to test your PRINT instruction data during the application development.

The following example prints the message to port 2. We use a PD contact, which causes the message instruction to be active for just one scan. Note the \$N at the end of the message, which produces a carriage return / line feed on the printer. This prepares the printer to print the next line, starting from the left margin.



**V-memory element** - this is used for printing V-memory contents in the integer format or real format. Use V-memory number or V-memory number with ":" and data type. The data types are shown in the table below. The Character code must be capital letters.



**NOTE:** There must be a space entered before and after the V-memory address to separate it from the text string. Failure to do this will result in an error code 499.

#	Character code	Description
1	none	16-bit binary (decimal number)
2	: B	4 digit BCD
3	: D	32-bit binary (decimal number)
4	: D B	8 digit BCD

Example:

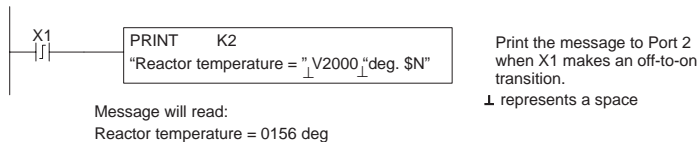
V2000 Print binary data in V2000 for decimal number

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

**Example:** The following example prints a message containing text and a variable. The "reactor temperature" labels the data, which is at V2000. You can use the : B qualifier after the V2000 if the data is in BCD format, for example. The final string adds the units of degrees to the line of text, and the \$N adds a carriage return / line feed.



**V-memory text element** "This is used for printing text stored in V-memory. Use the % followed by the number of characters after V-memory number for representing the text. If you assign "0" as the number of characters, the print function will read the character count from the first location. Then it will start at the next V-memory location and read that number of ASCII codes for the text from memory.

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 C100 in ON/OFF format

V2000.15 : BOOL Prints the status of bit 15 in V2000 in TRUE/FALSE format

The maximum numbers of characters you can print is 128. The number of characters for each element is listed in the table below:

Element Type	Maximum Characters
Text, 1 character	1
16 bit binary	6
32 bit binary	11
4 digit BCD	4
8 digit BCD	8
Floating point (real number)	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.

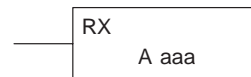
Special relay flags SP116 and SP117 indicate the status of the DL06 CPU ports (busy, or communications error). See the appendix on special relays for a description.



**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.

## Read from Network (RX)

The Read from Network instruction is used by the master device on a network to read a block of data from a slave device on the same network. The function parameters are loaded into the first and second level of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Read from Network function.



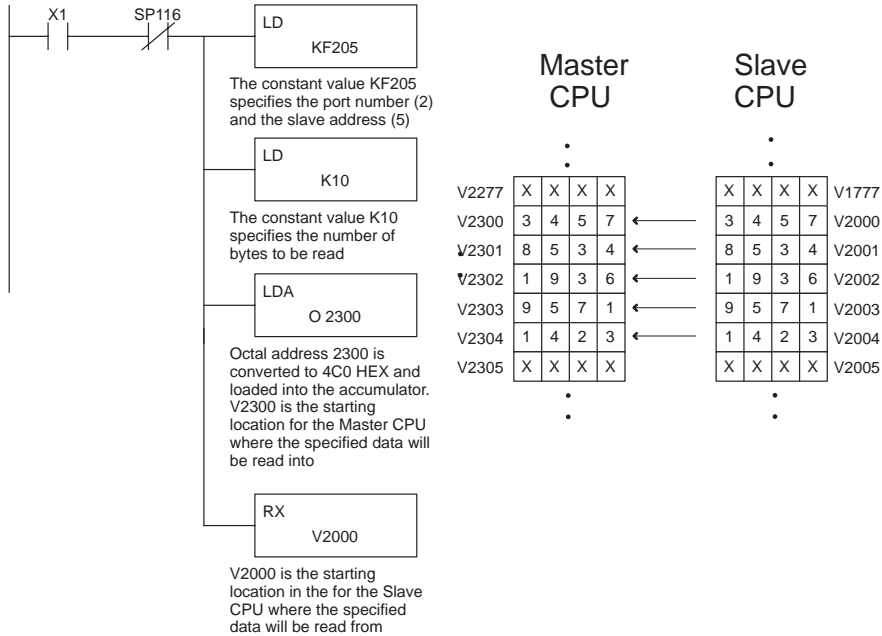
- Step 1: Load the slave address (0-- 90 BCD) into the first byte and the PLC internal port (KF2) or slot number of the master DCM or ECOM (0-- 7) into the second byte of the second level of the accumulator stack.
- Step 2: Load the number of bytes to be transferred into the first level of the accumulator stack.
- Step 3: Load the address of the data to be read into the accumulator. This parameter requires a HEX value.
- Step 4: Insert the RX instruction which specifies the starting Vmemory location (Aaaa) where the data will be read from in the slave.

Helpful Hint: — For parameters that require HEX values, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

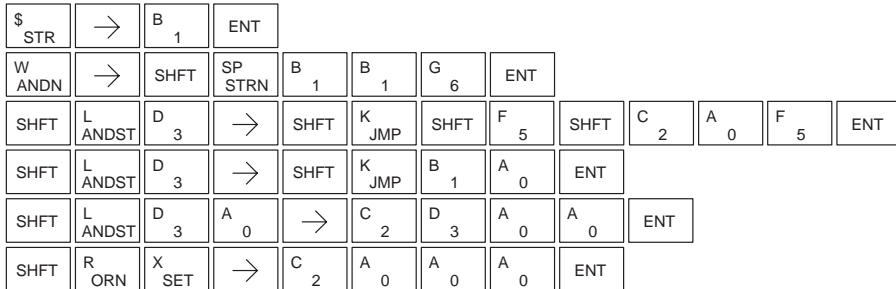
Operand Data Type	DL06 Range
..... <b>A</b>	<b>aaa</b>
V memory ..... V	See memory map
Pointer ..... P	See memory map
Inputs ..... X	0-777
Outputs ..... Y	0-777
Control Relays ..... C	0-1777
Stage ..... S	0-1777
Timer ..... T	0-377
Counter ..... CT	0-177
Special Relay ..... SP	0-777
Program Memory ..... \$	0-7680 (2K program mem.)

In the following example, when X1 is on and the port busy relay SP116 (see special relays) is not on, the RX instruction will access port 2 operating as a master. Ten consecutive bytes of data (V2000 – V2004) will be read from a CPU at station address 5 and copied into V memory locations V2300–V2304 in the CPU with the master port.

Direct SOFT32

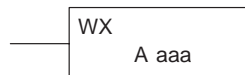


## Handheld Programmer Keystrokes



## Write to Network (WX)

The Write to Network instruction is used to write a block of data from the master device to a slave device on the same network. The function parameters are loaded into the accumulator and the first and second level of the stack. Listed below are the program steps necessary to execute the Write to Network function.



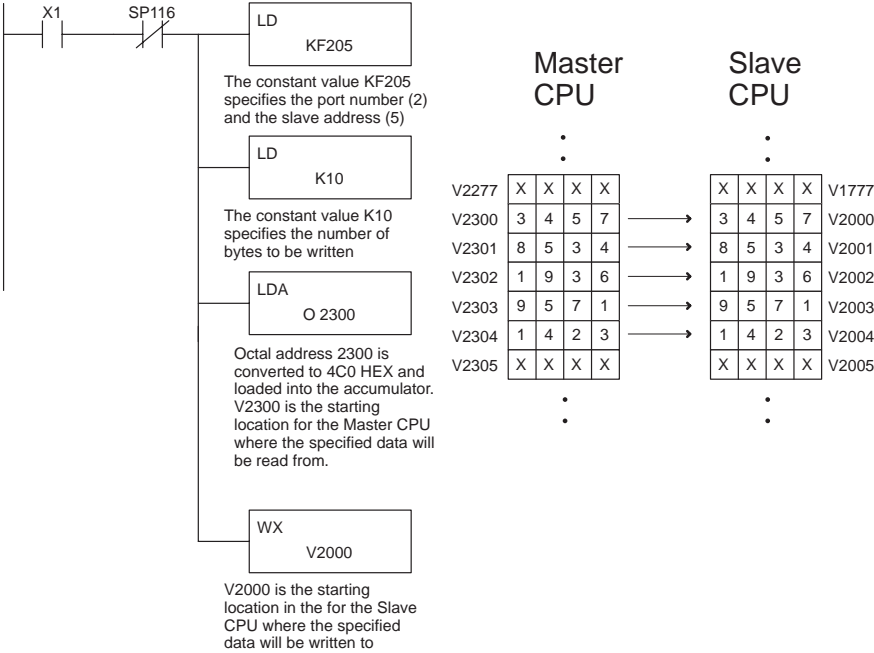
- Step 1: Load the slave address (0–90 BCD) into the low byte and “F2” into the high byte of the accumulator (the next two instructions push this word down to the second layer of the stack).
- Step 2: Load the number of bytes to be transferred into the accumulator (the next instruction pushes this word onto the top of the stack).
- Step 3: Load the starting Master CPU address into the accumulator. This is the memory location where the data will be written from. This parameter requires a HEX value.
- Step 4: Insert the WX instruction which specifies the starting V memory location (Aaaa) where the data will be written to in the slave.

**Helpful Hint:** — For parameters that require HEX values, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

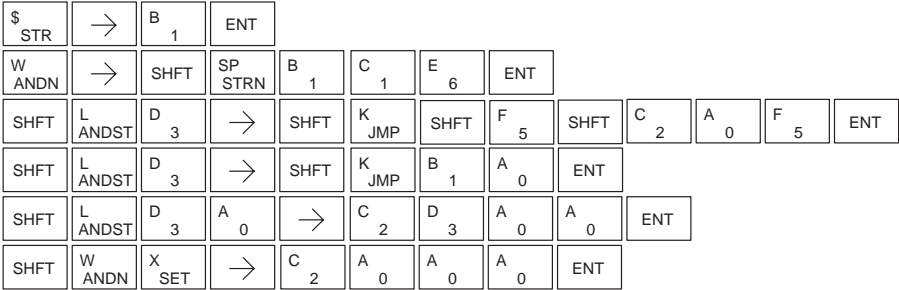
Operand Data Type	DL06 Range
..... <b>A</b>	<b>aaa</b>
V memory ..... V	See memory map
Pointer ..... P	See memory map
Inputs ..... X	0–777
Outputs ..... Y	0–777
Control Relays ..... C	0–1777
Stage ..... S	0–1777
Timer ..... T	0–377
Counter ..... CT	0–177
Special Relay ..... SP	0–777
Program Memory ..... \$	0–7680 (2K program mem.)

In the following example when X1 is on and the module busy relay SP116 (see special relays) is not on, the WX instruction will access port 2 operating as a master. Ten consecutive bytes of data is read from the Master CPU and copied to V memory locations V2000–V2004 in the slave CPU at station address 5.

Direct SOFT32

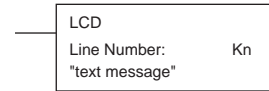


Handheld Programmer Keystrokes



## LCD

When enabled, the LCD instruction causes a user-defined text message to be displayed on the LCD Display Panel. The display is 16 characters wide by 2 rows high so a total of 32 characters can be displayed. Each row is addressed separately; the maximum number of characters the instruction will accept is 16.



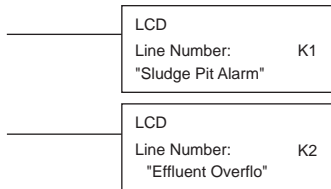
The text message can be entered directly into the message field of the instruction set-up dialog, or it can be located anywhere in user V-memory. If the text is located in V-memory, the LCD instruction is used to point to the memory location where the desired text originates. The length of the text string is also required.

From the DirectSOFT32 project folder, use the Instruction Browser to locate the LCD instruction. When you select the LCD instruction and click OK, the LCD dialog will appear, as shown in the examples. The LCD instruction is inserted into the ladder program via this set-up dialog box.

Display text strings can include embedded variables. Date and time settings and V-memory values can be embedded in the displayed text. Examples of each are shown.

### Direct Text Entry

The two dialogs to the right show the selections necessary to create the two ladder instructions below. Double quotation marks are required to delineate the text string. In the first dialog, the text “Sludge Pit Alarm” uses sixteen character spaces and will appear on line 1 when the instruction is enabled. Note, the line number is K1. Clicking the “check” button causes the instruction to be inserted into the ladder program.



By identifying the second Line Number as K2, the text string “Effluent Overflow” will appear on the second line of the display when the second instruction is enabled.

S	l	u	d	g	e		P	i	t		A	l	a	r	m
E	f	f	l	u	e	n	t		O	v	e	r	f	l	o

LCD

Line Number : K1

☒ LCD message

Message : "Sludge Pit Alarm"

☐ From V-memory

Starting V-memory address :

Number of characters :

LCD

Line Number : K2

☒ LCD message

Message : "Effluent Overflow"

☐ From V-memory

Starting V-memory address :

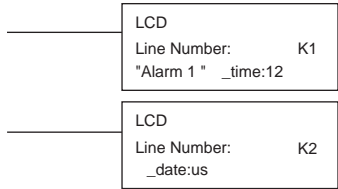
Number of characters :



Embedding date and/or time variables

The date and/or time can be embedded in the displayed text by using the variables listed in the table below. These variables can be included in the “LCD message” field of the LCD dialog. In the example the time variable (12 hour format) is embedded by adding `_time:12`. This time format uses a maximum of seven character spaces. The second dialog creates an instruction that prints the date on the second line of the display, when enabled.

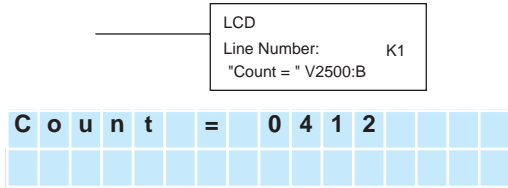
Date and Time Variables and Formats		
<code>_date:us</code>	US format	MM/DD/YY
<code>_date:e</code>	European format	DD/MM/YY
<code>_date:a</code>	Asian format	YY/MM/DD
<code>_time:12</code>	12 hour format	HH:MMAM/PM
<code>_time:24</code>	24 hour format	HH:MM:SS



A	l	a	r	m	1		1	1	:	2	1	P	M
0	5	-	0	8	-	0	2						

Embedding V-memory data

Any V-memory data can be displayed in any one of six available data formats. An example appears to the right. A list of data formats and modifiers is on the next page. Note that different data formats require differing numbers of character positions on the display.



LCD  
Line Number: K1  
☒ LCD message  
Message: "Alarm 1 " \_time:12  
☐ From V-memory  
Starting V-memory address:   
Number of characters:

LCD  
Line Number: K2  
☒ LCD message  
Message: \_date:us  
☐ From V-memory  
Starting V-memory address:   
Number of characters:

LCD  
Line Number: K1  
☒ LCD message  
Message: "Count = " V2500:B  
☐ From V-memory  
Starting V-memory address:   
Number of characters:

## Data Format Suffixes for Embedded V-memory Data

Several data formats are available for displaying V-memory data on the LCD. The choices are shown in the table below. A colon is used to separate the embedded V-memory location from the data format suffix and modifier. An example appears on the previous page.

Data Format	Modifier	Example	Displayed Characters													
<b>none (16-bit format)</b>		V2000 = 0000 0000 0001 0010	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>										
		V2000				1	8									
	[:S]	V2000:S	1	8												
	[:C0]	V2000:C0	0	0	1	8										
	[:0]	V2000:0			1	8										
<b>:B (4 digit BCD)</b>		V2000 = 0000 0000 0001 0010	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>										
	[:B]	V2000:B	0	0	1	2										
	[:BS]	V2000:BS	1	2												
	[:BC0]	V2000:BC0	0	0	1	2										
	[:B0]	V2000:B0			1	2										
<b>:D (32-bit decimal)</b>		V2000 = 0000 0000 0000 0000	<b>Double Word</b>													
		V2001 = 0000 0000 0000 0001	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>			
	[:D]	V2000:D							6	5	5	3	6			
	[:DS]	V2000:DS	6	5	5	3	6									
	[:DC0]	V2000:DC0	0	0	0	0	0	0	6	5	5	3	6			
	[:D0]	V2000:D0							6	5	5	3	6			
<b>:DB (8 digit BCD)</b>		V2000 = 0000 0000 0000 0000	<b>Double Word</b>													
		V2001 = 0000 0000 0000 0011	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>						
	[:DB]	V2000:DB	0	0	0	3	0	0	0	0						
	[:DBS]	V2000:DBS	3	0	0	0	0									
	[:DBC0]	V2000:DBC0	0	0	0	3	0	0	0	0						
	[:DB0]	V2000:DB0				3	0	0	0	0						
<b>:R (DWord floating point number)</b>		V2001/V2000 = 222.11111 (real number)	<b>Double Word</b>													
	[:R]	V2000:R				f	2	2	2	.	1	1	1	1	1	
	[:RS]	V2000:RS	f	2	2	2	.	1	1	1	1	1				
	[:RC0]	V2000:RC0	f	0	0	0	2	2	2	.	1	1	1	1	1	
	[:R0]	V2000:R0				f	2	2	2	.	1	1	1	1	1	
<b>:E (DWord floating point number with exponent)</b>		V2001/V2000 = 222.1 (real number)	<b>Double Word</b>													
	[:E]	V2000:E		f	2	.	2	2	1	0	0	E	+	0	2	
	[:ES]	V2000:ES	f	2	.	2	2	1	0	0	E	+	0	2		
	[:EC0]	V2000:EC0	f	2	.	2	2	1	0	0	E	+	0	2		
	[:E0]	V2000:E0	f	2	.	2	2	1	0	0	E	+	0	2		

f = plus/minus flag (plus = no symbol, minus = -)

The S, C0, and 0 modifiers alter the presentation of leading zeros and spaces. S removes leading spaces and left justifies the result. C0 replaces leading spaces with leading zeros. 0 is a modification of C0. 0 eliminates any leading zeros in the C0 format version and converts them to spaces.

## Text Entry from V-memory

Alternatively, text that resides in V-memory can be displayed on the LCD following the example on this page. The LCD dialog is used twice, once for each line on the display. The dialog requires the address of the first character to be displayed and the number of characters to be displayed.

For example, the two dialogs shown on this page would create the two LCD instructions below. When enabled, these instructions would cause the ASCII characters in V10000 - V10020 to be displayed. The ASCII characters and their corresponding memory locations are shown in the table below.

LCD

Line Number : K1

☐ LCD message

Message :

☐ From V-memory

Starting V-memory address : V10000

Number of characters : K16

LCD

Line Number : K2

☐ LCD message

Message :

☒ From V-memory

Starting V-memory address : V10010

Number of characters : K16

LCD

Line Number: K1

Starting V Memory Address: V10000

Number of Characters: K16

LCD

Line Number: K2

Starting V Memory Address: V10010

Number of Characters: K16

A	d	m	i	n		O	f	f	i	c	e				
H	i	g	h			T	e	m	p		A	l	a	r	m

V10000	d	A
V10001	i	m
V10002		n
V10003	f	O
V10004	i	f
V10005	e	c
V10006		
V10007		
V10010	i	H
V10011	h	g
V10012	T	
V10013	m	e
V10014		p
V10015	l	A
V10016	r	a
V10017		m

## MODBUS RTU Instructions

### MODBUS Read from Network (MRX)

The MODBUS Read from Network (MRX) instruction is used by the DL06 network master to read a block of data from a connected slave device and to write the data into V-memory addresses within the master. The instruction allows the user to specify the MODBUS Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, MODBUS data format and the Exception Response Buffer.

- Port Number: must be DL06 Port 2 (K2)
- Slave Address: specify a slave station address (0–247)
- Function Code: The following MODBUS function codes are supported by the MRX instruction:
  - 01 – Read a group of coils
  - 02 – Read a group of inputs
  - 03 – Read holding registers
  - 04 – Read input registers
  - 07 – Read Exception status
- Start Slave Memory Address: specifies the starting slave memory address of the data to be read. See the table on the following page.
- Start Master Memory Address: specifies the starting memory address in the master where the data will be placed. See the table on the following page.
- Number of Elements: specifies how many coils, inputs, holding registers or input register will be read. See the table on the following page.
- MODBUS Data Format: specifies MODBUS 584/984 or 484 data format to be used
- Exception Response Buffer: specifies the master memory address where the Exception Response will be placed. See the table on the following page.

### MRX Slave Address Ranges

Function Code	MODBUS Data Format	Slave Address Range(s)
01 – Read Coil	484 Mode	1–999
01 – Read Coil	584/984 Mode	1–65535
02 – Read Input Status	484 Mode	1001–1999
02 – Read Input Status	584/984 Mode	10001–19999 (5 digit) or 100001–165535 (6 digit)
03 – Read Holding Register	484 Mode	4001–4999
03 – Read Holding Register	584/984 Mode	40001–49999 (5 digit) or 4000001–465535 (6 digit)
04 – Read Input Register	484 Mode	3001–3999
04 – Read Input Register	584/984 Mode	30001–39999 (5 digit) or 3000001–365535 (6 digit)
07 – Read Exception Status	484 and 584/984 Mode	n/a

### MRX Master Memory Address Ranges

Operand Data Type	DL06 Range
Inputs ..... X	0–1777
Outputs ..... Y	0–1777
Control Relays..... C	0–3777
Stage Bits ..... S	0–1777
Timer Bits ..... T	0–377
Counter Bits ..... CT	0–377
Special Relays ..... SP	0–777
V-memory ..... V	all
Global Inputs ..... GX	0–3777
Global Outputs ..... GY	0–3777

### Number of Elements

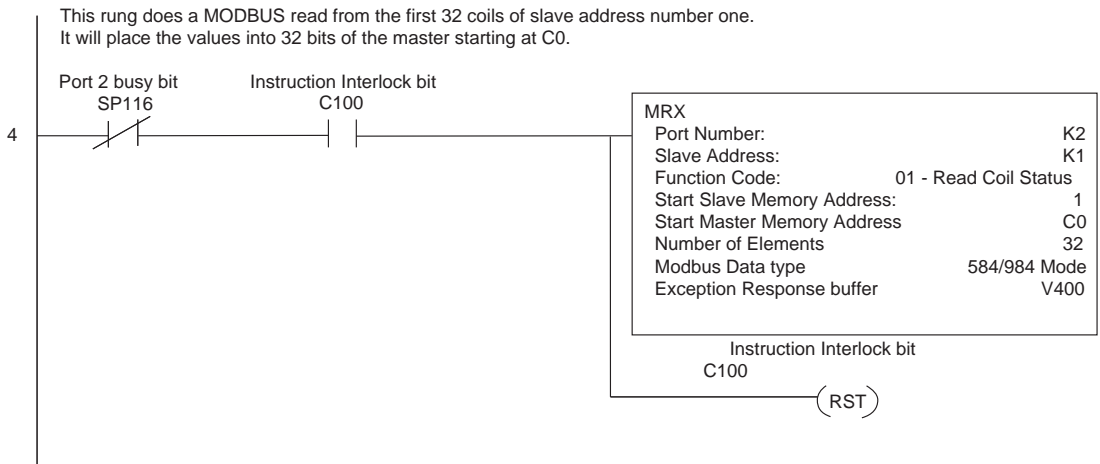
Operand Data Type	DL06 Range
V-memory ..... V	all
Constant ..... K	Bits: 1–2000 Registers: 1–125

### Exception Response Buffer

Operand Data Type	DL06 Range
V-memory ..... V	all

## MRX Example

DL06 port 2 has two Special Relay contacts associated with it (see Appendix D for comm port special relays). One indicates "Port busy" (SP116), and the other indicates "Port Communication Error" (SP117). The "Port Busy" bit is on while the PLC communicates with the slave. When the bit is off the program can initiate the next network request. The "Port Communication Error" bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an MRX or MWX instruction is executed. Typically network communications will last longer than 1 CPU scan. The program must wait for the communications to finish before starting the next transaction.



### MODBUS Write to Network (MWX)

The MODBUS Write to Network (MWX) instruction is used to write a block of data from the network master's (DL06) memory to MODBUS memory addresses within a slave device on the network. The instruction allows the user to specify the MODBUS Function Code, slave station address, starting master and slave memory addresses, number of elements to transfer, MODBUS data format and the Exception Response Buffer.

- Port Number: must be DL06 Port 2 (K2)
- Slave Address: specify a slave station address (0–247)
- Function Code: The following MODBUS function codes are supported by the MWX instruction:
  - 05 – Force Single coil
  - 06 – Preset Single Register
  - 15 – Force Multiple Coils
  - 16 – Preset Multiple Registers
- Start Slave Memory Address: specifies the starting slave memory address where the data will be written.
- Start Master Memory Address: specifies the starting address of the data in the master that is to be written to the slave.
- Number of Elements: specifies how many consecutive coils or registers will be written to. This field is only active when either function code 15 or 16 is selected.
- MODBUS Data Format: specifies MODBUS 584/984 or 484 data format to be used
- Exception Response Buffer: specifies the master memory address where the Exception Response will be placed

## MWX Slave Address Ranges

MWX Slave Address Ranges		
Function Code	MODBUS Data Format	Slave Address Range(s)
05 – Force Single Coil	484 Mode	1–999
05 – Force Single Coil	584/984 Mode	1–65535
06 – Preset Single Register	484 Mode	4001–4999
06 – Preset Single Register	584/984 Mode	40001–49999 (5 digit) or 400001–465535 (6 digit)
15 – Force Multiple Coils	484 Mode	1–999
15 – Force Multiple Coils	585/984 Mode	1–65535
16 – Preset Multiple Registers	484 Mode	4001–4999
16 – Preset Multiple Registers	584/984 Mode	40001–49999 (5 digit) or 4000001–465535 (6 digit)

## MWX Master Memory Address Ranges

MWX Master Memory Address Ranges	
Operand Data Type	DL06 Range
Inputs ..... X	0–1777
Outputs ..... Y	0–1777
Control Relays ..... C	0–3777
Stage Bits ..... S	0–1777
Timer Bits ..... T	0–377
Counter Bits ..... CT	0–377
Special Relays ..... SP	0–777
V–memory ..... V	all
Global Inputs ..... GX	0–3777
Global Outputs ..... GY	0–3777

## MWX Number of Elements

Number of Elements	
Operand Data Type	DL06 Range
V–memory ..... V	all
Constant ..... K	Bits: 1–2000 Registers: 1–125

## MWX Exception Response Buffer

Number of Elements	
Operand Data Type	DL06 Range
V–memory ..... V	all

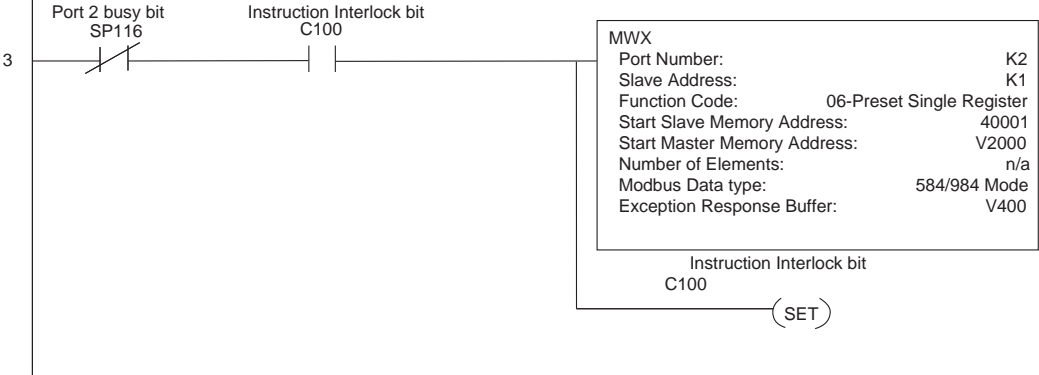


MWX Example

DL06 port 2 has two Special Relay contacts associated with it (see Appendix D for comm port special relays). One indicates “Port busy”(SP116), and the other indicates “Port Communication Error”(SP117). The “Port Busy” bit is on while the PLC communicates with the slave. When the bit is off the program can initiate the next network request. The “Port Communication Error” bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an MRX or MWX instruction is executed.

Typically network communications will last longer than 1 CPU scan. The program must wait for the communications to finish before starting the next transaction.

This rung does a MODBUS write to the first holding register 40001 of slave address number one. It will write the values over that reside in V2000. This particular function code only writes to 1 register. Use Function Code 16 to write to multiple registers. Only one Network instruction (WX, RX, MWX, MRX) can be enabled in one scan. That is the reason for the interlock bits. For using many network instructions on the same port, look at using the Shift Register instruction.



## ASCII Instructions

The DL06 CPU supports several instructions and methods that allow ASCII strings to be read into and written from the PLC communications ports. Specifically, port 2 on the DL06 can be used for either reading or writing raw ASCII strings, but cannot be used for both on the same CPU. The DL06 can also decipher ASCII embedded within a supported protocol (K-Sequence, DirectNet, Modbus) via the CPU port.

### Reading ASCII Input Strings

There are several methods that the DL06 can use to read ASCII input strings.

- 1) ASCII IN (AIN) – This instruction configures port 2 for raw ASCII input strings with parameters such as fixed and variable length ASCII strings, termination characters, byte swapping options, and instruction control bits. Use barcode scanners, weight scales, etc. to write raw ASCII input strings into port 2 based on the (AIN) instruction's parameters.
- 2) Write embedded ASCII strings directly to V-memory from an external HMI or similar master device via a supported communications protocol using the CPU ports. The AIN instruction is not used in this case. 3) If a DL06 PLC is a master on a network, the Network Read instruction (RX) can be used to read embedded ASCII data from a slave device via a supported communications protocol using port 2. The RX instruction places the data directly into V-memory.

### Writing ASCII Output Strings

The following instructions can be used to write ASCII output strings:

- 1) Print from V-memory (PRINTV) – Use this instruction to write raw ASCII strings out of port 2 to a display panel or a serial printer, etc. The instruction features the starting V-memory address, string length, byte swapping options, etc. When the instruction's permissive bit is enabled, the string is written to port 2.
- 2) Print to V-memory (VPRINT) – Use this instruction to create pre-coded ASCII strings in the PLC (i.e. alarm messages). When the instruction's permissive bit is enabled, the message is loaded into a pre-defined V-memory address location. Then the (PRINTV) instruction may be used to write the pre-coded ASCII string out of port 2. American, European and Asian Time/Date stamps are supported.

Additionally, if a DL06 PLC is a master on a network, the Network Write instruction (WX) can be used to write embedded ASCII data to an HMI or slave device directly from V-memory via a supported communications protocol using port 2.

### Managing the ASCII Strings

The following instructions can be helpful in managing the ASCII strings within the CPUs V-memory:

- ASCII Find (AFIND) – Finds where a specific portion of the ASCII string is located in continuous V-memory addresses. Forward and reverse searches are supported.
- ASCII Extract (AEX) – Extracts a specific portion (usually some data value) from the ASCII find location or other known ASCII data location.
- Compare V-memory (CMPV) – This instruction is used to compare two blocks of V-memory addresses and is usually used to detect a change in an ASCII string. Compared data types must be of the same format (i.e. BCD, ASCII, etc.).
- Swap Bytes (SWAPB) – usually used to swap V-memory bytes on ASCII data that was written directly to V-memory from an external HMI or similar master device via a communications protocol. The AIN and AEX instructions have a built-in byte swap feature.

## ASCII Input (AIN)

The ASCII Input instruction allows the CPU to receive ASCII strings through the specified communications port and places the string into a series of specified V-memory registers. The ASCII data can be received as a fixed number of bytes or as a variable length string with a specified termination character(s). Other features include, Byte Swap preferences, Character Timeout, and user defined flag bits for Busy, Complete and Timeout Error.

### AIN Fixed Length Configuration

- **Length Type:** select fixed length based on the length of the ASCII string that will be sent to the CPU port
- **Port Number:** must be DL06 port 2 (K2)
- **Data Destination:** specifies where the ASCII string will be placed in V-memory
- **Fixed Length:** specifies the length, in bytes, of the fixed length ASCII string the port will receive
- **Inter-character Timeout:** if the amount of time between incoming ASCII characters exceeds the set time, the specified Timeout Error bit will be set. No data will be stored at the Data Destination V-memory location. The bit will reset when the AIN instruction permissive bits are disabled. 0ms selection disables this feature.
- **First Character Timeout:** if the amount of time from when the AIN is enabled to the time the first character is received exceeds the set time, the specified First Character Timeout bit will be set. The bit will reset when the AIN instruction permissive bits are disabled. 0ms selection disables this feature.
- **Byte Swap:** swaps the high-byte and low-byte within each V-memory register of the Fixed Length ASCII string. See the SWAPB instruction for details.
- **Busy Bit:** is ON while the AIN instruction is receiving ASCII data
- **Complete Bit:** is set once the ASCII data has been received for the specified fixed length and reset when the AIN instruction permissive bits are disabled.
- **Inter-character Timeout Error Bit:** is set when the Character Timeout is exceeded. See Character Timeout explanation above.

AIN

Length Type

☒ Fixed Length  
☐ Variable Length

Port Number : K2

Data Destination : V2000

\* Data Destination = Byte count  
\* Data Destination + 1 = Start of data

Fixed Length : K32

Interchar. Timeout : 20 ms

First Char. Timeout : None

Byte Swap :

☐ None  
☒ All  
☐ All but null

Termination Code Length

☒ 1 Character  
☐ 2 Characters

TermCode 1 : hexadecimal

TermCode 2 : hexadecimal

Overflow Error :

Busy : C0

Complete : C1

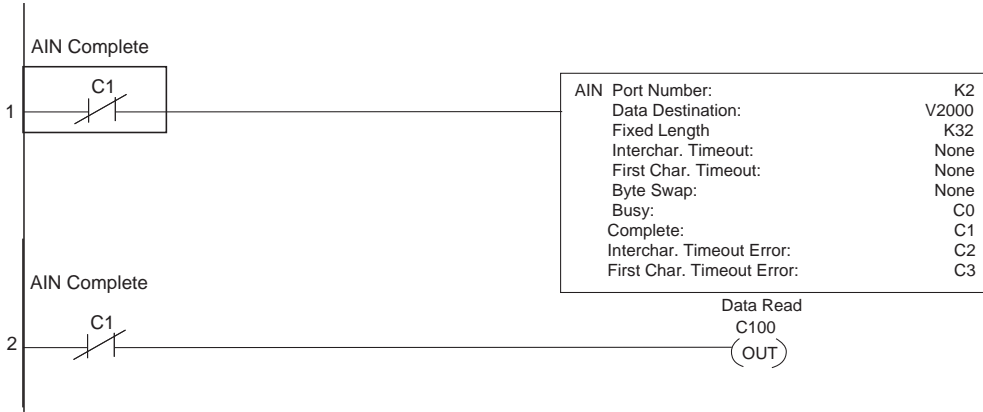
Interchar. T/O Error : C2

First Char. T/O Error : C3

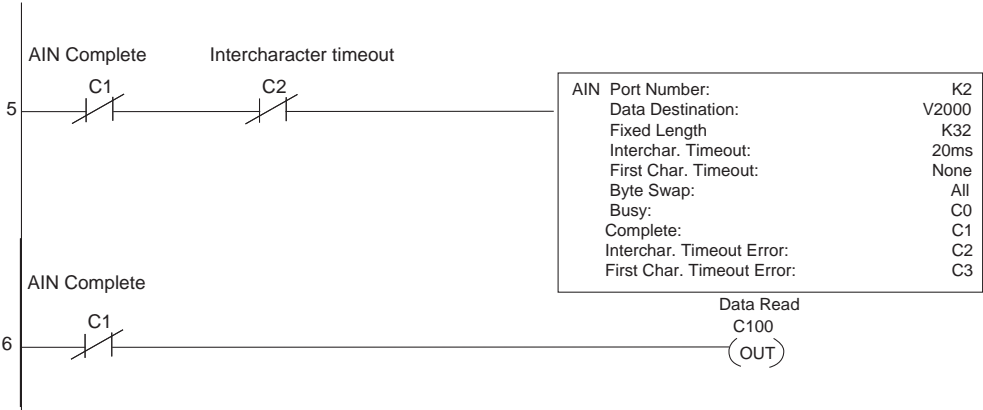
Parameter	
Data Destination	All V-memory
Fixed Length	K1-128
Bits: Busy, Complete, Timeout Error, Overflow	C0-3777

AIN Fixed Length Examples

Fixed Length example when the PLC is reading the port continuously and timing is not critical



Fixed Length example when character to character timing is critical



**AIN Variable Length Configuration:**

- Length Type: select Variable Length if the ASCII string length followed by termination characters will vary in length
- Port Number: must be DL06 port 2 (K2)
- Data Destination: specifies where the ASCII string will be placed in V-memory Maximum Variable Length: specifies, in bytes, the maximum length of a Variable Length ASCII string the port will receive
- Inter-character Timeout: if the amount of time between incoming ASCII characters exceeds the set time, the Timeout Error bit will be set. No data will be stored at the Data Destination V-memory location. The Timeout Error bit will reset when the AIN instruction permissive bits are disabled. 0ms selection disables this feature.
- First Character Timeout: if the amount of time from when the AIN is enabled to the time the first character is received exceeds the set time, the specified First Character Timeout bit will be set. The bit will reset when the AIN instruction permissive bits are disabled. 0ms selection disables this feature.
- Byte Swap: swaps the high-byte and low-byte within each V-memory register of the Variable Length ASCII string. See the SWAPB instruction for details.
- Termination Code Length: consists of either 1 or 2 characters. Refer to the ASCII table on the following page.
- Busy Bit: is ON while the AIN instruction is receiving ASCII data
- Complete Bit: is set once the ASCII data has been received up to the termination code characters. It will be reset when the AIN instruction permissive bits are disabled.
- Inter-character Timeout Error Bit: is set when the Character Timeout is exceeded. See Character Timeout explanation above.
- First Character Timeout Error Bit: is set when the First Character Timeout is exceeded. See First Character Timeout explanation above.
- Overflow Error Bit: is set when the ASCII data received exceeds the Maximum Variable Length specified.

AIN

Length Type

☐ Fixed Length

☒ Variable Length

Port Number : K2

Data Destination : V2000

\* Data Destination = Byte count  
\* Data Destination + 1 = Start of data

Maximum Variable Length : K40

Interchar. Timeout : 100 ms

First Char. Timeout : 2000 ms

Byte Swap :

☐ None

☐ All

☒ All but null

Termination Code Length

☒ 1 Character

☐ 2 Characters

TermCode 1 : 0D hexadecimal

TermCode 2 : 00 hexadecimal

Overflow Error : C4

Busy : C0

Complete : C1

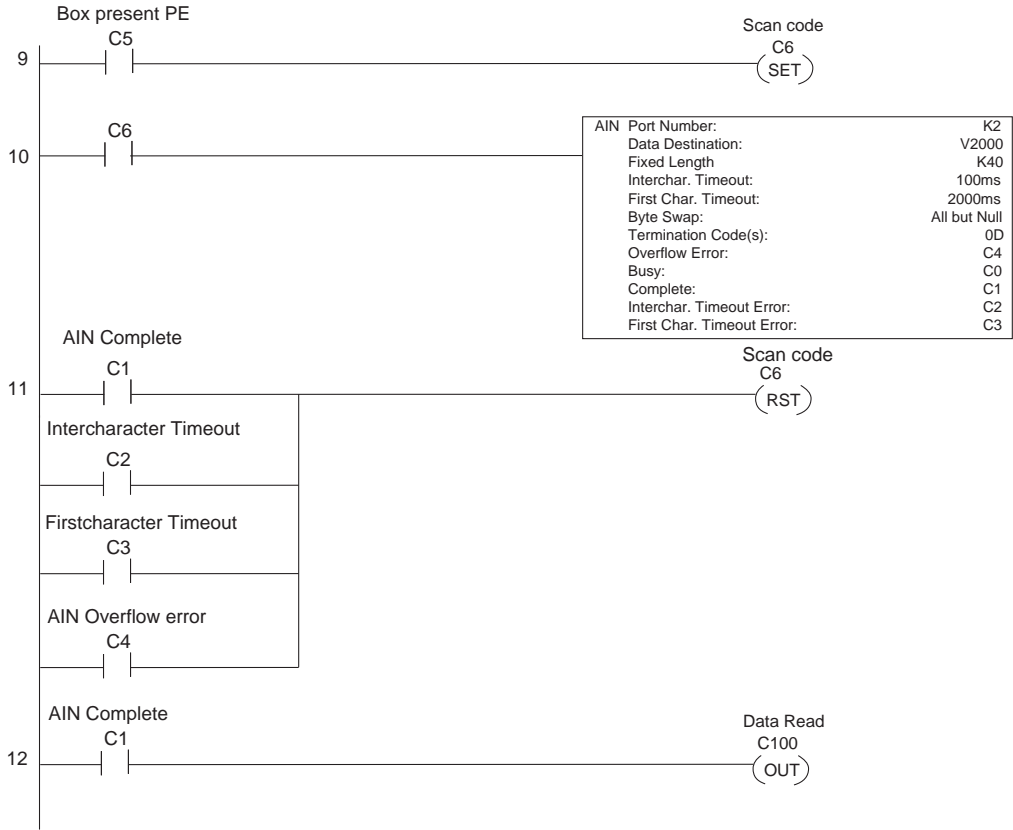
Interchar. T/O Error : C2

First Char. T/O Error : C3

Parameter	
Data Destination	All V-memory
Fixed Length	K1-128
Bits: Busy, Complete, Timeout Error, Overflow	C0-3777

AIN Variable Length Example

AIN variable length example used to read barcodes on boxes (PE = photoelectric sensor)



## ASCII Find (AFIND)

The ASCII Find instruction locates a specific ASCII string or portion of an ASCII string within a range of V-memory registers and places the string's Found Index number (byte number where desired string is found), in Hex, into a specified V-memory register. Other features include, Search Starting Index number for skipping over unnecessary bytes before beginning the FIND operation, Forward or Reverse direction search, and From Beginning and From End selections to reference the Found Index Value.

- **Base Address:** specifies the beginning V-memory register where the entire ASCII string is stored in memory
- **Total Number of Bytes:** specifies the total number of bytes to search for the desired ASCII string
- **Search Starting Index:** specifies which byte to skip to (with respect to the Base Address) before beginning the search
- **Direction:** Forward begins the search from lower numbered V-memory registers to higher numbered V-memory registers. Reverse does the search from higher numbered V-memory registers to lower numbered V-memory registers.
- **Found Index Value:** specifies whether the Beginning or the End byte of the ASCII string found will be loaded into the Found Index register
- **Found Index:** specifies the V-memory register where the Found Index Value will be stored. A value of FFFF will result if the desired string is not located in the memory registers specified.
- **Search for String:** up to 128 characters.

Parameter	DL06 Range
Base Address	All V-memory
Total Number of Bytes	All V-memory or K1-128
Search Starting Index	All V-memory or K0-127
Found	Index All V-memory



**NOTE:** Quotation marks are not required around the Search String item. Quotes are valid characters that the AFIND can search for.



## AFIND Search Example

In the following example, the AFIND instruction is used to search for the “day” portion of “Friday” in the ASCII string “Today is Friday.”, which had previously been loaded into V-memory. Note that a Search Starting Index of constant (K) 5 combined with a Forward Direction Search is used to prevent finding the “day” portion of the word “Today”. The Found Index will be placed into V4000.

[illegible]

Notice that quotation marks are not placed around the Search String. Only use quotation marks if they're actually part of the Search String.

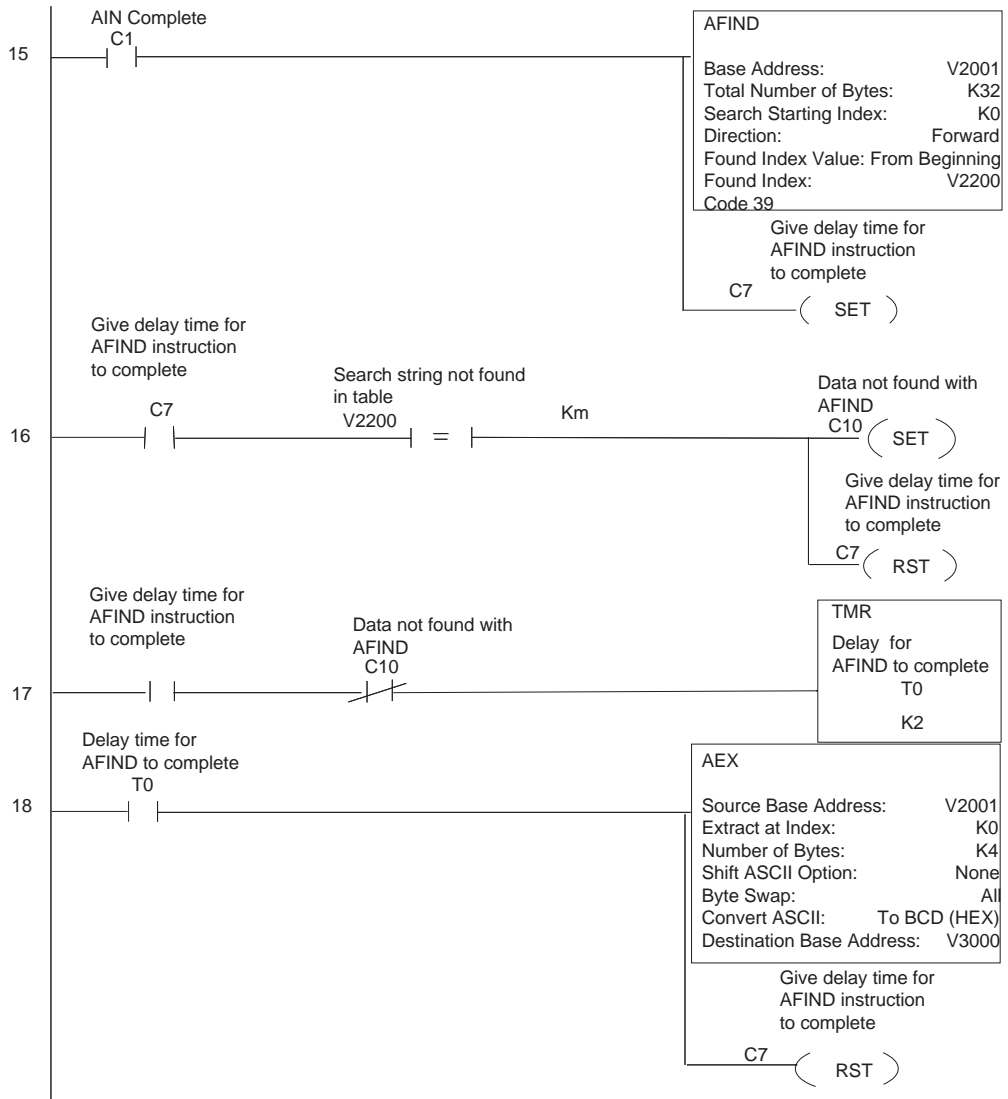
Diagram illustrating a memory search process. The search starts at index 5 (Search start Index Number) and moves in the Reverse Direction Search (upwards) to index 0. From index 0, it moves in the Forward Direction Search (downwards) to index 12 (Beginning Index Number). The search continues to index 13 (End Index Number) and finally finds the character 'd' at index 12 (Found Index Number = 0012).

Base Address	ASCII Characters	HEX Equivalent	Search Status
0	T	54h	Low
1	o	6Fh	High
2	d	64h	Low
3	a	61h	High
4	y	79h	Low
5		20h	High
6	i	69h	Low
7	s	73h	High
8		20h	Low
9	F	46h	High
10	r	72h	Low
11	i	69h	High
12	d	64h	Low
13	a	61h	High
14	y	79h	Low
15	.	2Eh	High

Found Index Number = 0012

## AFIND Example Combined with AEX Instruction

When an AIN instruction has executed, its Complete bit can be used to trigger an AFIND instruction to search for a desired portion of the ASCII string. Once the string is found, the AEX instruction can be used to extract the located string.



### ASCII Extract (AEX)

The ASCII Extract instruction extracts a specified number of bytes of ASCII data from one series of V-memory registers and places it into another series of V-memory registers. Other features include, Extract at Index for skipping over unnecessary bytes before beginning the Extract operation, Shift ASCII Option, for One Byte Left or One Byte Right, Byte Swap and Convert data to a BCD format number.

- **Source Base Address:** specifies the beginning V-memory register where the entire ASCII string is stored in memory
- **Extract at Index:** specifies which byte to skip to (with respect to the Source Base Address) before extracting the data
- **Number of Bytes:** specifies the number of bytes to be extracted
- **Shift ASCII Option:** shifts all extracted data one byte left or one byte right to displace “unwanted” characters if necessary
- **Byte Swap:** swaps the high-byte and the low-byte within each V-memory register of the extracted data. See the SWAPB instruction for details.
- **Convert BCD(Hex) ASCII to BCD (Hex):** if enabled, this will convert ASCII numerical characters to Hexadecimal numerical values
- **Destination Base Address:** specifies the V-memory register where the extracted data will be stored

See the previous page for an example using the AEX instruction.

Parameter	DL06 Range	
<b>Source Base Address</b>	All V-memory	
<b>Extract at Index</b>	All V-memory or K0-127	
<b>Number of Bytes</b> “Convert BCD (HEX) ASCII” not checked	Constant range: K1-128	V-memory location containing BCD value: 1-128
<b>Number of Bytes</b> “Convert BCD (HEX) ASCII” checked	Constant range: K1-4	V-memory location containing BCD value: 1-4
<b>Destination Base Address</b>	All V-memory	

**AEX**

Source Base Address : V4500

Extract at Index : V4200

Number of Bytes : K8

Shift ASCII Option : ☒ None ☐ One Byte Left ☐ One Byte Right

Byte Swap : ☒ None ☐ All ☐ All but Null

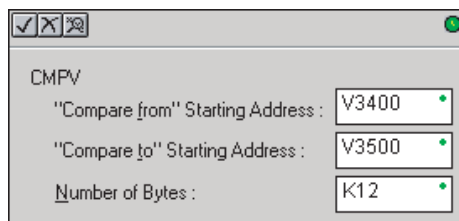
☐ Convert BCD(HEX)ASCII to BCD(HEX)

Destination Base Address : V4100

## ASCII Compare (CMPV)

The ASCII Compare instruction compares two groups of V-memory registers. The CMPV will compare any data type (ASCII to ASCII, BCD to BCD, etc.) of one series (group) of V-memory registers to another series of V-memory registers for a specified byte length.

- “Compare from” Starting Address: specifies the beginning V-memory register of the first group of V-memory registers to be compared from.
- “Compare to” Starting Address: specifies the beginning V-memory register of the second group of V-memory registers to be compared to.
- Number of Bytes: specifies the length of each V-memory group to be compared

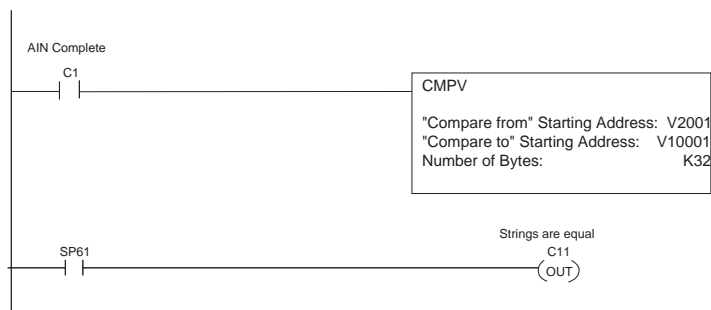


SP61 = 1, the result is equal  
SP61 = 0, the result is not equal

Parameter	DL06 Range
Compare from Starting Address	All V-memory
Compare to Starting Address	All V-memory
Number of Bytes	K0-127

### CMPV Example

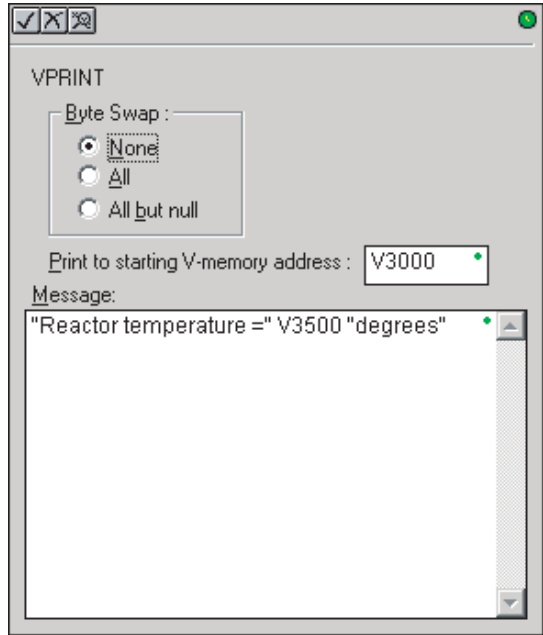
The CMPV instruction executes when the AIN instruction is complete. If the compared V-memory tables are equal, SP61 will turn ON.



## ASCII Print to V-memory (VPRINT)

The ASCII Print to V-memory instruction will write a specified ASCII string into a series of V-memory registers. Other features include Byte Swap, options to suppress or convert leading zeros or spaces, and \_Date and \_Time options for U.S., European, and Asian date formats and 12 or 24 hour time formats.

- **Byte Swap:** swaps the high-byte and low-byte within each V-memory register the ASCII string is printed to. See the SWAPB instruction for details.
- **Print to Starting V-memory Address:** specifies the beginning of a series of V-memory addresses where the ASCII string will be placed by the VPRINT instruction.
- **Starting V-memory Address:** the first V-memory register of the series of registers specified will contain the ASCII string's length in bytes.
- **Starting V-memory Address +1:** the 2nd and subsequent registers will contain the ASCII string printed to V-memory.



Parameter	DL06 Range
Print to Starting V-memory Address	All V-memory

VPRINT Time / Date Stamping– the codes in the table below can be used in the VPRINT ASCII string message to “print to V-memory” the current time and/or date.

#	Character code	Date / Time Stamp Options
1	_Date:us	American standard (month/day/2 digit year)
2	_Date:e European standard	(day/month/2 digit year)
3	_Date:a Asian standard	(2 digit year/month/day)
4	_Time:12	standard 12 hour clock (0–12 hour:min am/pm)
5	_Time:24	standard 24 hour clock (0–12 hour:min am/pm)

**VPRINT V-memory element** – the following modifiers can be used in the VPRINT ASCII string message to “print to V-memory” register contents in integer format or real format. Use V-memory number or V-memory number with “:” and data type. The data types are shown in the table below. The Character code must be capital letters.



*NOTE: There must be a space entered before and after the V-memory address to separate it from the text string. Failure to do this will result in an error code 499.*

#	Character code	Description
1	none	16-bit binary (decimal number)
2	: B	4 digit BCD
3	: D	32-bit binary (decimal number)
4	: D B	8 digit BCD
5	: R	Floating point number (real number)
6	: E	Floating point number (real number with exponent)

Examples:

V2000 Print binary data in V2000 for decimal number

V2000 : B Print BCD data in V2000

V2000 : D Print binary number in V2000 and V2001 for decimal number

V2000 : D B Print BCD data in V2000 and V2001

V2000 : R Print floating point number in V2000/V2001 as real number

V2000 : E Print floating point number in V2000/V2001 as real number with exponent

The following modifiers can be added to any of the modifies above to suppress or convert leading zeros or spaces. The character code must be capital letters.

#	Character code	Description
1	S	Suppresses leading spaces
2	C0	Converts leading spaces to zeros
3	0	Suppresses leading zeros

Example with V2000 = 0018 (binary format)

V-memory Register with Modifier	Number of Characters			
	1	2	3	4
V2000	0	0	1	8
V2000:B	0	0	1	2
V2000:BO	1	2		

Example with V2000 = sp sp18 (binary format) where sp = space

V-memory Register with Modifier	Number of Characters			
	1	2	3	4
V2000	sp	sp	1	8
V2000:B	sp	sp	1	2
V2000:BS	1	2		
V2000:BC0	0	0	1	2

**VPRINT V-memory text element** – the following is used for “printing to V-memory” text stored in registers. Use the % followed by the number of characters after V-memory number for representing the text. If you assign “0” as the number of characters, the function will read the character count from the first location. Then it will start at the next V-memory location and read that number of ASCII codes for the text from memory.

Example:

V2000 % 16 16 characters in V2000 to V2007 are printed.

V2000 % 0 The characters in V2001 to Vxxxx (determined by the number in V2000) will be printed.

**VPRINT Bit element** – the following is used for “printing to V-memory” the state of the designated bit in V-memory or a control relay bit. The bit element can be assigned by the designating point (.) and bit number preceded by the V-memory number or relay number. The output type is described as shown in the table below.

#	Data format	Description
1	none	Print 1 for an ON state, and 0 for an OFF state
2	: BOOL	Print “TRUE” for an ON state, and “FALSE” for an OFF state
3	: ONOFF	Print “ON” for an ON state, and “OFF” for an OFF state

Example:

V2000 . 15 Prints the status of bit 15 in V2000, in 1/0 format

C100 Prints the status of C100 in 1/0 format

C100 : BOOL Prints the status of C100 in TRUE/FALSE format

C100 : ON/OFF Prints the status of 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 VPRINT is 128. The number of characters required for each element, regardless of whether the :S, :C0 or :0 modifiers are used, is listed in the table below.

Element type	Maximum Characters
Text, 1 character	1
16 bit binary	6
32 bit binary	11
4 digit BCD	4
8 digit BCD	8
Floating point (real number)	3
Floating point (real with exponent)	13
V-memory/text	2
Bit (1/0 format)	1
Bit (TRUE/FALSE format)	5
Bit (ON/OFF format)	3

**Text element** – the following is used for “printing to V-memory” character strings. The character strings are defined as the character (more than 0) ranged by the double quotation marks. Two hex numbers preceded by the dollar sign means an 8-bit ASCII character code. Also, two characters preceded by the dollar sign is interpreted according to the following table:

#	Character code	Description
1	\$\$	Dollar sign (\$)
2	\$"	Double quotation (")
3	\$Lor \$I	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.

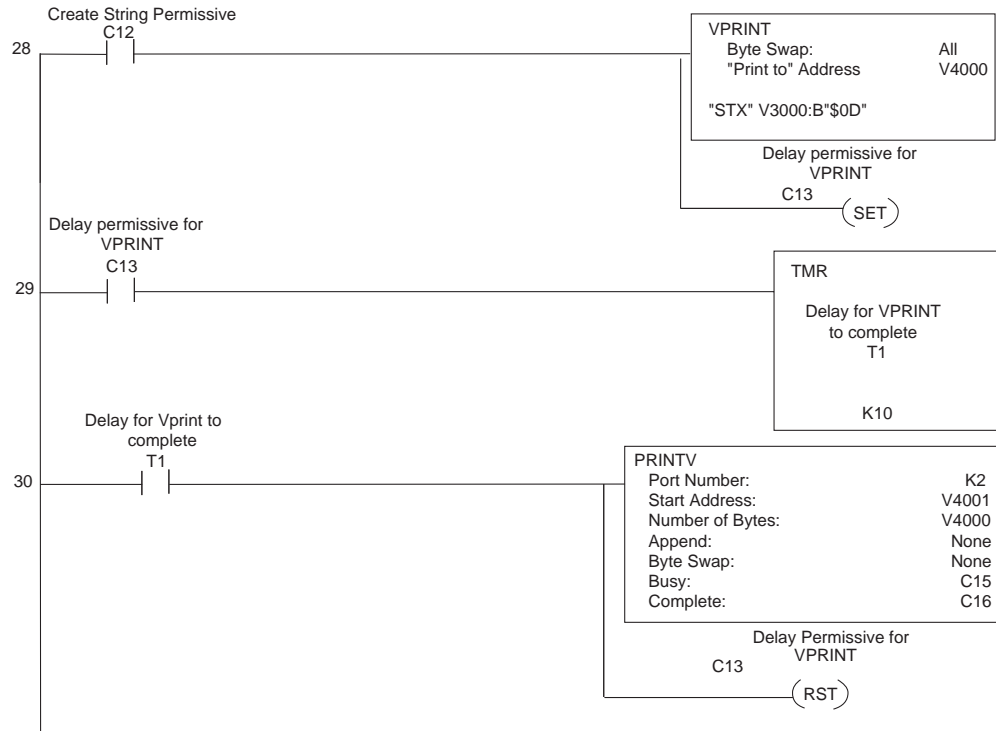
" "	Length 0 without character
"A"	Length 1 with character A
" "	Length 1 with blank
"\$"	Length 1 with double quotation mark
"\$R\$L"	Length 2 with one CR and one LF
"\$OD\$OA"	Length 2 with one CR and one LF
"\$\$"	Length 1 with one \$ mark

In printing an ordinary line of text, you will need to include double quotation marks before and after the text string. Error code 499 will occur in the CPU when the print instruction contains invalid text or no quotations. It is important to test your VPRINT instruction data during the application development.



VPRINT Example Combined with PRINTV Instruction

The VPRINT instruction is used to create a string in V-memory. The PRINTV is used to print the string out of port 2.



## ASCII Print from V-memory (PRINTV)

The ASCII Print from V-memory instruction will send an ASCII string out of the designated communications port from a specified series of V-memory registers for a specified length in number of bytes. Other features include user specified Append Characters to be placed after the desired data string for devices that require specific termination character(s), Byte Swap options, and user specified flags for Busy and Complete.

- Port Number: must be DL06 port 2 (K2)
- Start Address: specifies the beginning of series of V-memory registers that contain the ASCII string to print
- Number of Bytes: specifies the length of the string to print
- Append Characters: specifies ASCII characters to be added to the end of the string for devices that require specific termination characters
- Byte Swap: swaps the high-byte and low-byte within each V-memory register of the string while printing. See the SWAPB instruction for details.
- Busy Bit: will be ON while the instruction is printing ASCII data
- Complete Bit: will be set once the ASCII data has been printed and reset when the PRINTV instruction permissive bits are disabled.

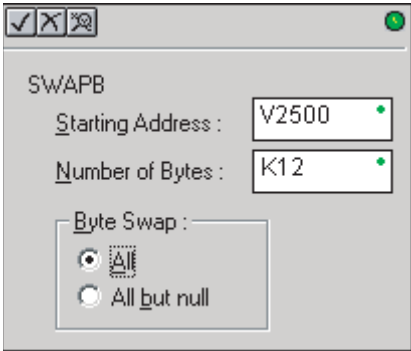
See the previous page for an example using the PRINTV instruction.

Parameter	DL06 Range
Port Number	port 2 (K2)
Start Address	All V-memory
Number of Bytes	All V-memory or k1-128
Bits: Busy, Complete	C0-3777

ASCII Swap Bytes (SWAPB)

The ASCII Swap Bytes instruction swaps byte positions (high-byte to low-byte and low-byte to high-byte) within each V-memory register of a series of V-memory registers for a specified number of bytes.

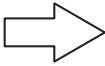
- Starting Address: specifies the beginning of a series of V-memory registers the instruction will use to begin byte swapping
- Number of Bytes: specifies the number of bytes, beginning with the Starting Address, to byte swap.



Parameter	DL06 Range
Starting Address	All V-memory
Number of Bytes	All V-memory or K1-128

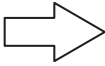
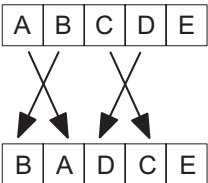
Byte Swap Preferences

No Byte Swapping  
(AIN, AEX, PRINTV, VPRINT)



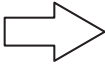
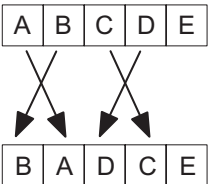
	Byte	
	High	Low
V2000	0005h	
V2001	B	A
V2002	D	C
V2003	xx	E

Byte Swap All



	Byte	
	High	Low
V2000	0005h	
V2001	A	B
V2002	C	D
V2003	E	xx

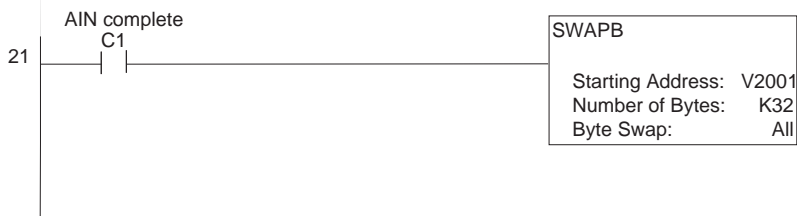
Byte Swap All but Null



	Byte	
	High	Low
V2000	0005h	
V2001	A	B
V2002	C	D
V2003	xx	E

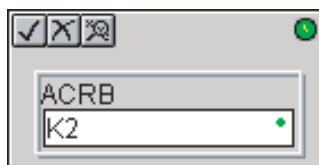
### SWAPB Example

The AIN Complete bit is used to trigger the SWAPB instruction. Use a one-shot so the SWAPB only executes once.



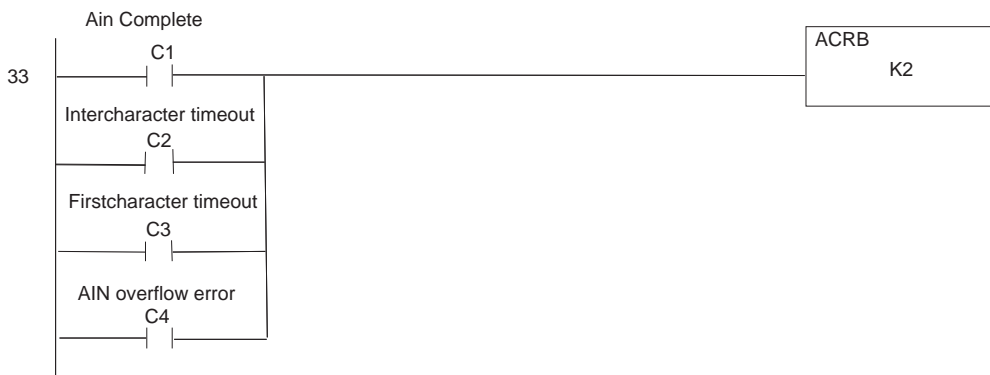
### ASCII Clear Buffer (ACRB)

The ASCII Clear Buffer instruction will clear the ASCII receive buffer of the specified communications port number. Port Number: must be DL06 port 2 (K2)



### ACRB Example

The AIN Complete bit or the AIN diagnostic bits are used to clear the ASCII buffer.



Informações sobre programação  
[www.soliton.com.br](http://www.soliton.com.br) - e-mail: [soliton@soliton.com.br](mailto:soliton@soliton.com.br)

**SOLITON CONTROLES INDUSTRIAIS LTDA**

Rua Alfredo Pujol, 1010 - Santana - São Paulo - SP.

Tel: 11 - 6950-1834 / Fax: 11 - 6979-8980 - e-mail: [vendas@soliton.com.br](mailto:vendas@soliton.com.br)