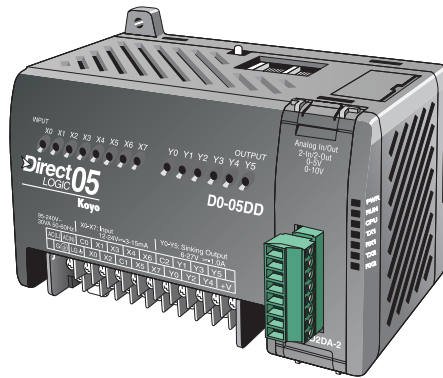




# DL05 User Manual

Manual Number: D0-USER-M

## Volume 1 of 2



## ⚡ WARNING ⚡

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

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

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

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

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

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

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

## Trademarks

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

**Copyright 2013, AutomationDirect.co Incorporated  
All Rights Reserved**

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

## ⚡ AVERTISSEMENT ⚡

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

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

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

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

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

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

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

## Marques de commerce

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

**Copyright 2013, Automationdirect.co Incorporated**  
**Tous droits réservés**

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

# DL05 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-USER-M  
**Issue:** Sixth Edition, Rev. C  
**Issue Date:** 2/13

Publication History		
Issue	Date	Description of Changes
Original	12/98	Original issue
2nd Edition	2/00	added PID chapter, analog module chapter and memory cartridge chapter
2nd Edition, Rev. A	7/00	added DC power
3rd Edition	11/01	removed MC and analog module chapters, corrected drum instruction, several minor corrections, added PLC weights, EU directive additions
3rd Edition, Rev. A	7/02	Added new discrete option modules
4th Edition	11/02	Converted manual to QuarkXPress
5th Edition	6/04	Removed option module data, added MC chapter, updated instruction set, inserted memory appendix, made minor corrections
6th Edition	12/08	Corrected E-stop, updated instruction set, added <b>DirectSOFT</b> 5 IBox instructions to Chapter 5, revised PID chapter, moved HSIO chapter to Appendices, divided Chapter 4 into Chapters 3 & 4, added Numbering Systems to Appendix section, made corrections throughout manual
6th Edition, Rev. A	4/10	Made minor corrections throughout manual.
6th Edition, Rev. B	8/11	Corrected number of registers needed in the print message instruction. Corrected TIME instruction: changed CPU to read Memory Cartridge. Made other minor corrections throughout manual.
6th Edition, Rev. C	2/13	Added H0-CTRIO2 references Updated suppression for inductive loads Made minor corrections throughout manual.

# VOLUME ONE:

# TABLE OF CONTENTS

---



## Chapter 1: Getting Started

<b>Introduction</b> .....	1-2
The Purpose of this Manual .....	1-2
Where to Begin .....	1-2
Supplemental Manuals .....	1-2
Technical Support .....	1-2
<b>Conventions Used</b> .....	1-3
Key Topics for Each Chapter .....	1-3
<b>DL05 Micro PLC Components</b> .....	1-4
The DL05 Micro PLC Family .....	1-4
<i>Direct</i> SOFT 5 Programming for Windows™ .....	1-4
Handheld Programmer .....	1-5
<b>I/O Selection Quick Chart</b> .....	1-5
<b>Quick Start for PLC Checkout and Programming</b> .....	1-6
<b>Steps to Designing a Successful System</b> .....	1-10
<b>Questions and Answers about DL05 Micro PLCs</b> .....	1-12

## Chapter 2: Installation, Wiring, and Specifications

<b>Safety Guidelines</b> .....	2-2
Plan for Safety .....	2-2
Three Levels of Protection .....	2-3
Emergency Stops .....	2-3
Emergency Power Disconnect .....	2-4
Orderly System Shutdown .....	2-4
Class 1, Division 2 Approval .....	2-4
<b>Orientation to DL05 Front Panel</b> .....	2-5
Connector Removal .....	2-6

## Table of Contents

---

<b>Mounting Guidelines</b> .....	<b>2-7</b>
Unit Dimensions .....	2-7
Enclosures .....	2-7
Panel Layout & Clearances .....	2-8
Using DIN Rail Mounting Rails .....	2-9
Environmental Specifications .....	2-10
Agency Approvals .....	2-10
Marine Use .....	2-10
<b>Wiring Guidelines</b> .....	<b>2-11</b>
Fuse Protection for Input Power .....	2-11
External Power Source .....	2-12
Planning the Wiring Routes .....	2-12
Fuse Protection for Input and Output Circuits .....	2-13
I/O Point Numbering .....	2-13
<b>System Wiring Strategies</b> .....	<b>2-14</b>
PLC Isolation Boundaries .....	2-14
Connecting Operator Interface Devices .....	2-15
Connecting Programming Devices .....	2-15
Sinking/Sourcing Concepts .....	2-16
I/O “Common” Terminal Concepts .....	2-17
Connecting DC I/O to Solid State Field Devices .....	2-18
Solid State Input Sensors .....	2-18
Solid State Output Loads .....	2-18
Relay Output Wiring Methods .....	2-20
Relay Outputs-Transient Suppression for Inductive Loads in a Control System ...	2-21
Prolonging Relay Contact Life .....	2-26
DC Input Wiring Methods .....	2-28
DC Output Wiring Methods .....	2-29
High Speed I/O Wiring Methods .....	2-30
<b>Wiring Diagrams and Specifications</b> .....	<b>2-32</b>
D0-05AR I/O Wiring Diagram .....	2-32
D0-05DR I/O Wiring Diagram .....	2-34
D0-05AD I/O Wiring Diagram .....	2-36
D0-05DD I/O Wiring Diagram .....	2-38
D0-05AA I/O Wiring Diagram .....	2-40

D0–05DA I/O Wiring Diagram .....	2–42
D0–05DR-D I/O Wiring Diagram .....	2–44
D0–05DD–D I/O Wiring Diagram .....	2–46
<b>Glossary of Specification Terms .....</b>	<b>2–48</b>

## Chapter 3: CPU Specifications and Operation

<b>Introduction .....</b>	<b>3–2</b>
DL05 CPU Features .....	3–2
<b>CPU Specifications .....</b>	<b>3–3</b>
<b>CPU Hardware Setup .....</b>	<b>3–4</b>
Communication Port Pinout Diagrams .....	3–4
Connecting the Programming Devices .....	3–5
CPU Setup Information .....	3–5
Status Indicators .....	3–6
Mode Switch Functions .....	3–6
Changing Modes in the DL05 PLC .....	3–7
Mode of Operation at Power-up .....	3–7
Auxiliary Functions .....	3–8
Clearing an Existing Program .....	3–8
Initializing System Memory .....	3–8
Setting Retentive Memory Ranges .....	3–9
Using a Password .....	3–10
<b>CPU Operation .....</b>	<b>3–11</b>
CPU Operating System .....	3–11
Program Mode .....	3–12
Run Mode .....	3–12
Read Inputs .....	3–13
Service Peripherals and Force I/O .....	3–13
Update Special Relays and Special Registers .....	3–14
Solve Application Program .....	3–14
Write Outputs .....	3–15
Diagnostics .....	3–15
<b>I/O Response Time .....</b>	<b>3–15</b>
Is Timing Important for Your Application? .....	3–15
Normal Minimum I/O Response .....	3–15

## Table of Contents

---

Normal Maximum I/O Response .....	3-16
Improving Response Time .....	3-17
<b>CPU Scan Time Considerations .....</b>	<b>3-18</b>
Reading Inputs .....	3-18
Writing Outputs .....	3-18
Application Program Execution .....	3-19
PLC Numbering Systems .....	3-20
PLC Resources .....	3-20
V-memory .....	3-21
Binary-Coded Decimal Numbers .....	3-21
Hexadecimal Numbers .....	3-21
<b>Memory Map .....</b>	<b>3-22</b>
Octal Numbering System .....	3-22
Discrete and Word Locations .....	3-22
V-memory Locations for Discrete Memory Areas .....	3-22
Input Points (X Data Type) .....	3-23
Output Points (Y Data Type) .....	3-23
Control Relays (C Data Type) .....	3-23
Timers and Timer Status Bits (T Data Type) .....	3-23
Timer Current Values (V Data Type) .....	3-24
Counters and Counter Status Bits (CT Data type) .....	3-24
Counter Current Values (V Data Type) .....	3-24
Word Memory (V Data Type) .....	3-25
Stages (S Data type) .....	3-25
Special Relays (SP Data Type) .....	3-25
<b>DL05 System V-memory .....</b>	<b>3-26</b>
System Parameters and Default Data Locations (V Data Type) .....	3-26
DL05 Memory Map Table .....	3-28
<b>DL05 Aliases .....</b>	<b>3-29</b>
<b>X Input Bit Map .....</b>	<b>3-30</b>
<b>Y Output Bit Map .....</b>	<b>3-30</b>
<b>Control Relay Bit Map .....</b>	<b>3-31</b>
<b>Stage Control/Status Bit Map .....</b>	<b>3-32</b>
<b>Timer Status Bit Map .....</b>	<b>3-32</b>
<b>Counter Status Bit Map .....</b>	<b>3-33</b>



## Chapter 4: Configuration and Connections

<b>DL05 System Design Strategies</b> .....	<b>4-2</b>
I/O System Configurations .....	4-2
Networking Configurations .....	4-2
Automatic I/O Configuration .....	4-3
Power Budgeting .....	4-3
<b>Network Configuration and Connections</b> .....	<b>4-4</b>
Configuring the DL05's Comm Ports .....	4-4
DL05 Port Specifications .....	4-4
Networking DL05 to DL05 RS-232C .....	4-4
Networking Using RS-422 Converters .....	4-5
Modbus Port Configuration .....	4-6
<i>DirectNET</i> Port Configuration .....	4-7
<b>Network Slave Operation</b> .....	<b>4-8</b>
Modbus Function Codes Supported .....	4-8
Determining the Modbus Address .....	4-8
If Your Host Software Requires the Data Type and Address...	4-9
Example 1: V2100 .....	4-10
Example 2: Y20 .....	4-10
Example 3: T10 Current Value .....	4-10
Example 4: C54 .....	4-10
If Your Modbus Host Software Requires an Address ONLY .....	4-11
Example 1: V2100 584/984 Mode .....	4-13
Example 2: Y20 584/984 Mode .....	4-13
Example 3: T10 Current Value 484 Mode .....	4-13
Example 4: C54 584/984 Mode .....	4-13
<b>Network Master Operation</b> .....	<b>4-14</b>
Step 1: Identify Master Port # and Slave # .....	4-15
Step 2: Load Number of Bytes to Transfer .....	4-15
Step 3: Specify Master Memory Area .....	4-16
Step 4: Specify Slave Memory Area .....	4-16
Communications from a Ladder Program .....	4-17
Multiple Read and Write Interlocks .....	4-17

## Chapter 5: Standard RLL and Standard RLL

<b>Introduction</b> .....	5-2
Instruction List .....	5-2
<b>Using Boolean Instructions</b> .....	5-4
END Statement .....	5-4
Simple Rungs .....	5-4
Normally Closed Contact .....	5-5
Contacts in Series .....	5-5
Midline Outputs .....	5-5
Parallel Elements .....	5-6
Joining Series Branches in Parallel .....	5-6
Joining Parallel Branches in Series .....	5-6
Combination Networks .....	5-6
Comparative Boolean .....	5-7
Boolean Stack .....	5-7
Immediate Boolean .....	5-8
<b>Boolean Instructions</b> .....	5-9
<b>Comparative Boolean</b> .....	5-25
<b>Immediate Instructions</b> .....	5-31
<b>Timer, Counter and Shift Register Instructions</b> .....	5-35
Using Timers .....	5-35
Timer Example Using Discrete Status Bits .....	5-37
Timer Example Using Comparative Contacts .....	5-37
Accumulating Fast Timer (TMRAF) .....	5-38
Accumulating Timer Example using Discrete Status Bits .....	5-39
Accumulator Timer Example Using Comparative Contacts .....	5-39
Using Counters .....	5-40
Counter Example Using Discrete Status Bits .....	5-42
Counter Example Using Comparative Contacts .....	5-42
Stage Counter Example Using Discrete Status Bits .....	5-44
Stage Counter Example Using Comparative Contacts .....	5-44
Up / Down Counter Example Using Discrete Status Bits .....	5-46
Up / Down Counter Example Using Comparative Contacts .....	5-46
<b>Accumulator/Stack Load and Output Data Instructions</b> .....	5-48

Using the Accumulator	5-48
Copying Data to the Accumulator	5-48
Changing the Accumulator Data	5-49
Using the Accumulator Stack	5-50
Using Pointers	5-51
<b>Logical Instructions (Accumulator)</b>	<b>5-60</b>
<b>Math Instructions</b>	<b>5-68</b>
<b>Bit Operation Instructions</b>	<b>5-82</b>
<b>Number Conversion Instructions (Accumulator)</b>	<b>5-87</b>
Shuffle Digits Block Diagram	5-94
<b>Table Instructions</b>	<b>5-96</b>
Copy Data From a Data Label Area to V-memory	5-98
<b>CPU Control Instructions</b>	<b>5-99</b>
<b>Program Control Instructions</b>	<b>5-101</b>
Understanding Master Control Relays	5-106
MLS/MLR Example	5-107
<b>Interrupt Instructions</b>	<b>5-108</b>
External Interrupt Program Example	5-109
Timed Interrupt Program Example	5-110
Independent Timed Interrupt	5-110
<b>Message Instructions</b>	<b>5-111</b>
Fault Example	5-111
Data Label Example	5-113
<b>Intelligent I/O Instructions</b>	<b>5-118</b>
<b>Network Instructions</b>	<b>5-120</b>
<b>Intelligent Box (IBox) Instructions</b>	<b>5-124</b>
(IBox) Instructions List	5-124

# VOLUME TWO:

# TABLE OF CONTENTS

---



## Chapter 6: Drum Instruction Programming

<b>DL05 Drum Introduction</b> .....	<b>6-2</b>
Purpose .....	6-2
Drum Terminology .....	6-2
Drum Chart Representation .....	6-3
Output Sequences .....	6-3
<b>Step Transitions</b> .....	<b>6-4</b>
Drum Instruction Types .....	6-4
Timer-Only Transitions .....	6-4
Timer and Event Transitions .....	6-5
Event-Only Transitions .....	6-6
Counter Assignments .....	6-6
Last Step Completion .....	6-7
<b>Overview of Drum Operation</b> .....	<b>6-8</b>
Drum Instruction Block Diagram .....	6-8
Powerup State of Drum Registers .....	6-9
<b>Drum Control Techniques</b> .....	<b>6-10</b>
Drum Control Inputs .....	6-10
Self-Resetting Drum .....	6-11
Initializing Drum Outputs .....	6-11
Using Complex Event Step Transitions .....	6-11
<b>Drum Instruction</b> .....	<b>6-12</b>
Timed Drum with Discrete Outputs (DRUM) .....	6-12
<b>Event Drum (EDRUM) Instruction</b> .....	<b>6-14</b>
Program Using the Handheld Programmer .....	6-16

## Chapter 7: RLL<sup>PLUS</sup> Stage Programming

<b>Introduction to Stage Programming</b> .....	7-2
Overcoming “Stage Fright” .....	7-2
<b>Learning to Draw State Transition Diagrams</b> .....	7-3
Introduction to Process States .....	7-3
The Need for State Diagrams .....	7-3
A 2-State Process .....	7-3
RLL Equivalent .....	7-4
Stage Equivalent .....	7-4
Let’s Compare .....	7-5
Initial Stages .....	7-5
What Stage Bits Do .....	7-6
Stage Instruction Characteristics .....	7-6
<b>Using the Stage Jump Instruction for State Transitions</b> .....	7-7
Stage Jump, Set, and Reset Instructions .....	7-7
<b>Stage Program Example: Toggle On/Off Lamp Controller</b> .....	7-8
A 4-State Process .....	7-8
<b>Four Steps to Writing a Stage Program</b> .....	7-9
<b>Stage Program Example: A Garage Door Opener</b> .....	7-10
Garage Door Opener Example .....	7-10
Draw the Block Diagram .....	7-10
Draw the State Diagram .....	7-11
Add Safety Light Feature .....	7-12
Modify the Block Diagram and State Diagram .....	7-12
Using a Timer Inside a Stage .....	7-13
Add Emergency Stop Feature .....	7-14
Exclusive Transitions .....	7-14
<b>Stage Program Design Considerations</b> .....	7-15
Stage Program Organization .....	7-15
How Instructions Work Inside Stages .....	7-16
Using a Stage as a Supervisory Process .....	7-17
Stage Counter .....	7-17
Power Flow Transition Technique .....	7-18
Stage View in <i>DirectSOFT 5</i> .....	7-18

<b>Parallel Processing Concepts</b> .....	<b>7-19</b>
Parallel Processes .....	7-19
Converging Processes .....	7-19
Convergence Stages (CV) .....	7-19
Convergence Jump (CVJMP) .....	7-20
Convergence Stage Guidelines .....	7-20
<b>RLL<sup>PLUS</sup> (Stage) Instructions</b> .....	<b>7-21</b>
Stage (SG) .....	7-21
Initial Stage (ISG) .....	7-22
JUMP (JMP) .....	7-22
Not Jump (NJMP) .....	7-22
Converge Stage (CV) and Converge Jump (CVJMP) .....	7-23
<b>Questions and Answers about Stage Programming</b> .....	<b>7-25</b>

## Chapter 8: PID Loop Operation

<b>DL05 PID Control</b> .....	<b>8-2</b>
DL05 PID Control Features .....	8-2
<b>Introduction to PID Control</b> .....	<b>8-4</b>
What is PID Control? .....	8-4
<b>Introducing DL05 PID Control</b> .....	<b>8-6</b>
Process Control Definitions .....	8-8
<b>PID Loop Operation</b> .....	<b>8-9</b>
Position Form of the PID Equation .....	8-9
Reset Windup Protection .....	8-10
Freeze Bias .....	8-11
Adjusting the Bias .....	8-11
Step Bias Proportional to Step Change in SP .....	8-12
Eliminating Proportional, Integral or Derivative Action .....	8-12
Velocity Form of the PID Equation .....	8-12
Bumpless Transfer .....	8-13
Loop Alarms .....	8-13
Loop Operating Modes .....	8-14
Special Loop Calculations .....	8-14
<b>Ten Steps to Successful Process Control</b> .....	<b>8-16</b>

<b>PID Loop Setup</b> .....	<b>8-18</b>
Some Things to Do and Know Before Starting .....	8-18
PID Error Flags .....	8-18
Establishing the Loop Table Size and Location .....	8-18
Loop Table Word Definitions .....	8-20
PID Mode Setting 1 Bit Descriptions (Addr + 00) .....	8-21
PID Mode Setting 2 Bit Descriptions (Addr + 01) .....	8-22
Mode/Alarm Monitoring Word (Addr + 06) .....	8-23
Ramp/Soak Table Flags (Addr + 33) .....	8-23
Ramp/Soak Table Location (Addr + 34) .....	8-24
Ramp/Soak Table Programming Error Flags (Addr + 35) .....	8-24
Configure the PID Loop .....	8-25
<b>PID Loop Tuning</b> .....	<b>8-40</b>
Open-Loop Test .....	8-40
Manual Tuning Procedure .....	8-41
Auto Tuning Procedure .....	8-44
Use <i>DirectSOFT</i> 5 Data View with PID View .....	8-48
Open a New Data View Window .....	8-48
Open PID View .....	8-49
<b>Using the Special PID Features</b> .....	<b>8-51</b>
How to Change Loop Modes .....	8-51
Operator Panel Control of PID Modes .....	8-52
PLC Modes Effect on Loop Modes .....	8-52
Loop Mode Override .....	8-52
PV Analog Filter .....	8-53
Creating an Analog Filter in Ladder Logic .....	8-54
Use the <i>DirectSOFT</i> 5 Filter Intelligent Box Instruction .....	8-55
FilterB Example .....	8-55
<b>Ramp/Soak Generator</b> .....	<b>8-56</b>
Introduction .....	8-56
Ramp/Soak Table .....	8-57
Ramp/Soak Table Flags .....	8-59
Ramp/Soak Generator Enable .....	8-59
Ramp/Soak Controls .....	8-59
Ramp/Soak Profile Monitoring .....	8-60

## Table of Contents

---

Ramp/Soak Programming Errors .....	8-60
Testing Your Ramp/Soak Profile .....	8-60
<b>DirectSOFT 5 Ramp/Soak Example .....</b>	<b>8-61</b>
Setup the Profile in PID Setup .....	8-61
Program the Ramp/Soak Control in Relay Ladder .....	8-61
Test the Profile .....	8-62
<b>Cascade Control .....</b>	<b>8-63</b>
Introduction .....	8-63
Cascaded Loops in the DL05 CPU .....	8-64
Tuning Cascaded Loops .....	8-65
<b>Time-Proportioning Control .....</b>	<b>8-66</b>
On/Off Control Program Example .....	8-67
<b>Feedforward Control .....</b>	<b>8-68</b>
Feedforward Example .....	8-69
<b>PID Example Program .....</b>	<b>8-70</b>
Program Setup for the PID Loop .....	8-70
<b>Troubleshooting Tips .....</b>	<b>8-72</b>
<b>Glossary of PID Loop Terminology .....</b>	<b>8-74</b>
<b>Bibliography .....</b>	<b>8-76</b>

## Chapter 9: Maintenance and Troubleshooting

<b>Hardware System Maintenance .....</b>	<b>9-2</b>
Standard Maintenance .....	9-2
<b>Diagnostics .....</b>	<b>9-2</b>
Diagnostics .....	9-2
Fatal Errors .....	9-2
Non-fatal Errors .....	9-2
V-memory Error Code Locations .....	9-3
Special Relays (SP) Corresponding to Error Codes .....	9-3
DL05 Micro PLC Error Codes .....	9-4
Program Error Codes .....	9-5
<b>CPU Indicators .....</b>	<b>9-6</b>
RUN Indicator .....	9-7



CPU Indicator .....	9-7
<b>Communications Problems .....</b>	<b>9-7</b>
<b>I/O Point Troubleshooting .....</b>	<b>9-8</b>
Possible Causes .....	9-8
Some Quick Steps .....	9-8
Handheld Programmer Keystrokes Used to Test an Output Point .....	9-9
<b>Noise Troubleshooting .....</b>	<b>9-10</b>
Electrical Noise Problems .....	9-10
Reducing Electrical Noise .....	9-10
<b>Machine Startup and Program Troubleshooting .....</b>	<b>9-11</b>
Syntax Check .....	9-11
Special Instructions .....	9-12
Duplicate Reference Check .....	9-13
Run Time Edits .....	9-14
Forcing I/O Points .....	9-16
Bit Forcing with Direct Access .....	9-16

## Chapter 10: Memory Cartridge/Real Time Clock

<b>General Information about the D0-01MC .....</b>	<b>10-2</b>
Jumper Selects Write Enable or Disable .....	10-2
Low Battery Alert .....	10-2
Clock/Calendar .....	10-2
Specifications .....	10-2
New Ladder Instructions .....	10-2
Error Code Changes .....	10-2
<b>Setting the Write Enable/Disable Jumper .....</b>	<b>10-3</b>
Write Enable .....	10-3
Write Disable .....	10-3
<b>Plugging-in the Memory Cartridge .....</b>	<b>10-4</b>
Remove the Slot Cover .....	10-4
Insert the Memory Cartridge .....	10-4
<b>Software and Firmware Requirements .....</b>	<b>10-5</b>
How to Update Your <i>DirectSOFT</i> Programming Software .....	10-5
How to Update Your DL05 Firmware .....	10-5

## Table of Contents

---

<b>Naming the Memory Cartridge</b> .....	<b>10-6</b>
Up to 8 Alphanumeric Characters .....	10-6
Name is Retained in Cartridge Memory and Project Folder .....	10-6
<b>Setting the Time and Date</b> .....	<b>10-7</b>
<b>Memory Transfers</b> .....	<b>10-8</b>
CPU to MC .....	10-8
MC to CPU .....	10-8
<b>LED Indicator Lights</b> .....	<b>10-9</b>
<b>Password Protected Programs</b> .....	<b>10-9</b>
<b>Memory Map and Forwarding Range</b> .....	<b>10-10</b>
<b>Battery Back-up During AC Power Loss</b> .....	<b>10-11</b>
What if the Battery Dies? .....	10-11
Battery Type .....	10-11
Removing and Replacing the Battery .....	10-11
<b>Specifications and Agency Approvals</b> .....	<b>10-12</b>
<b>Clock/Calendar Instructions</b> .....	<b>10-13</b>
Date (DATE) .....	10-13
Time (TIME) .....	10-14
Move Memory Cartridge / Load Label (MOVMC) (LDLBL) .....	10-15
Copy Data From a Data Label Area to V-memory .....	10-16
Copy Data From V-memory to a Data Label Area .....	10-17
<b>Error Codes</b> .....	<b>10-18</b>

## Appendix A: Auxiliary Functions

<b>Introduction</b> .....	<b>A-2</b>
Purpose of Auxiliary Functions .....	A-2
Accessing AUX Functions via <i>DirectSOFT 5</i> .....	A-3
Accessing AUX Functions via the Handheld Programmer .....	A-3
<b>AUX 2* — RLL Operations</b> .....	<b>A-4</b>
AUX 21 Check Program .....	A-4
AUX 22 Change Reference .....	A-4
AUX 23 Clear Ladder Range .....	A-4
AUX 24 Clear Ladders .....	A-4

<b>AUX 3* — V-memory Operations</b> .....	<b>A-4</b>
AUX 31 Clear V-Memory .....	A-4
<b>AUX 4* — I/O Configuration</b> .....	<b>A-5</b>
AUX 41 Show I/O Configuration .....	A-5
<b>AUX 5* — CPU Configuration</b> .....	<b>A-5</b>
AUX 51 Modify Program Name .....	A-5
AUX 53 Display Scan Time .....	A-5
AUX 54 Initialize Scratchpad .....	A-5
AUX 55 Set Watchdog Timer .....	A-5
AUX 56 CPU Network Address .....	A-6
AUX 57 Set Retentive Ranges .....	A-6
AUX 58 Test Operations .....	A-6
AUX 59 Bit Override .....	A-7
AUX 5B Counter Interface Configuration .....	A-7
AUX 5D Select PLC Scan Mode .....	A-7
<b>AUX 6* — Handheld Programmer Configuration</b> .....	<b>A-8</b>
AUX 61 Show Revision Numbers .....	A-8
AUX 62 Beeper On/Off .....	A-8
AUX 65 Run Self Diagnostics .....	A-8
<b>AUX 7* — EEPROM Operations</b> .....	<b>A-8</b>
Transferrable Memory Areas .....	A-8
AUX 71 CPU to HPP EEPROM .....	A-8
AUX 72 HPP EEPROM to CPU .....	A-9
AUX 73 Compare HPP EEPROM to CPU .....	A-9
AUX 74 HPP EEPROM Blank Check .....	A-9
AUX 75 Erase HPP EEPROM .....	A-9
AUX 76 Show EEPROM Type .....	A-9
<b>AUX 8* — Password Operations</b> .....	<b>A-9</b>
AUX 81 Modify Password .....	A-9
AUX 82 Unlock CPU .....	A-10
AUX 83 Lock CPU .....	A-10

## Appendix B: DL05 Error Codes

<b>DL05 Error Codes</b> .....	<b>B-2</b>
-------------------------------	------------

### Appendix C: Instruction Execution Times

<b>Introduction</b> .....	<b>C-2</b>
V-memory Data Registers .....	C-2
V-memory Bit Registers .....	C-2
How to Read the Tables .....	C-2
<b>Instruction Execution Times</b> .....	<b>C-3</b>
Boolean Instructions .....	C-3
Comparative Boolean Instructions .....	C-4
Immediate Instructions .....	C-10
Timer, Counter and Shift Register .....	C-10
Accumulator Data Instructions .....	C-11
Logical Instructions .....	C-12
Math Instructions .....	C-12
Bit Instructions .....	C-14
Number Conversion Instructions .....	C-14
Table Instructions .....	C-14
CPU Control Instructions .....	C-15
Program Control Instructions .....	C-15
Interrupt Instructions .....	C-15
Network Instructions .....	C-15
Message Instructions .....	C-16
RLL <sup>PLUS</sup> Instructions .....	C-16
Drum Instructions .....	C-16
Word Bit Instructions .....	C-17

### Appendix D: Special Relays

<b>DL05 PLC Special Relays</b> .....	<b>D-2</b>
--------------------------------------	------------

### Appendix E: High-speed Input and Pulse Output Features

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

<b>Choosing the HSIO Operating Mode</b> .....	<b>E-4</b>
Understanding the Six Modes .....	E-4
Default Mode .....	E-4
Configuring the HSIO Mode .....	E-5
Configuring Inputs X0 – X2 .....	E-5
<b>Mode 10: High-Speed Counter</b> .....	<b>E-6</b>
Purpose .....	E-6
Functional Block Diagram .....	E-6
Wiring Diagram .....	E-7
Interfacing to Counter Inputs .....	E-7
Interfacing to Counter Outputs .....	E-7
Setup for Mode 10 .....	E-8
Presets and Special Relays .....	E-8
Preset Data Starting Location .....	E-9
Using Fewer than 24 Presets .....	E-9
Equal Relay Numbers .....	E-9
Calculating Your Preset Values .....	E-10
X Input Configuration .....	E-10
Writing Your Control Program .....	E-11
Program Example: Counter Without Preset .....	E-12
Counter With Presets Program Example .....	E-14
Counter With Preload Program Example .....	E-16
Troubleshooting Guide for Mode 10 .....	E-17
Symptom: The counter does not count. ....	E-17
Symptom: The counter counts but the presets do not function. ....	E-17
Symptom: The counter counts up but will not reset. ....	E-17
<b>Mode 20: Quadrature Counter</b> .....	<b>E-18</b>
Purpose .....	E-18
Functional Block Diagram .....	E-18
Quadrature Encoder Signals .....	E-18
Wiring Diagram .....	E-19
Interfacing to Encoder Outputs .....	E-19
Setup for Mode 20 .....	E-20
X Input Configuration .....	E-20
Writing Your Control Program .....	E-21

## Table of Contents

---

Quadrature Counter w/Preload Program Example .....	E-21
Counter Preload Program Example .....	E-23
Troubleshooting Guide for HSIO Mode 20 .....	E-23
Symptom: The counter does not count. ....	E-23
Symptom: The counter counts in the wrong direction .....	E-23
Symptom: The counter counts up and down but will not reset. ....	E-23
<b>Mode 30: Pulse Output .....</b>	<b>E-24</b>
Purpose .....	E-24
Functional Block Diagram .....	E-25
Wiring Diagram .....	E-26
Interfacing to Drive Inputs .....	E-26
Motion Control Profile Specifications .....	E-27
Physical I/O Configuration .....	E-27
Logical I/O Functions .....	E-27
Setup for Mode 30 .....	E-28
Profile / Velocity Select Register .....	E-28
Profile Parameter Table .....	E-29
Trapezoidal Profile .....	E-29
Registration Profile .....	E-29
Velocity Profile .....	E-29
Choosing the Profile Type .....	E-30
Trapezoidal Profile Defined .....	E-30
Registration and Home Search Profiles Defined .....	E-30
Velocity Profile Defined .....	E-30
<b>Trapezoidal Profile Operation .....</b>	<b>E-31</b>
Trapezoidal Profile Applications .....	E-31
Trapezoidal Profile Program Example .....	E-32
Preload Position Value .....	E-33
<b>Registration Profile Operation .....</b>	<b>E-34</b>
Registration Applications .....	E-34
Registration Profile Program Example .....	E-35
Home Search Program Example .....	E-37
<b>Velocity Profile Operation .....</b>	<b>E-39</b>
Velocity Profile Applications .....	E-39
Velocity Profile Program Example .....	E-40

Pulse Output Error Codes . . . . .	E-42
Troubleshooting Guide for HSIO Mode 30 . . . . .	E-42
Symptom: The stepper motor does not rotate. . . . .	E-42
Symptom: The motor turns in the wrong direction. . . . .	E-43
<b>Mode 40: High-Speed Interrupts . . . . .</b>	<b>E-44</b>
Purpose . . . . .	E-44
Functional Block Diagram . . . . .	E-44
Setup for Mode 40 . . . . .	E-45
Interrupts and the Ladder Program . . . . .	E-45
External Interrupt Timing Parameters . . . . .	E-46
Timed Interrupt Parameters . . . . .	E-46
X Input/Timed INT Configuration . . . . .	E-46
Independent Timed Interrupt . . . . .	E-46
External Interrupt Program Example . . . . .	E-47
Timed Interrupt Program Example . . . . .	E-48
<b>Mode 50: Pulse Catch Input . . . . .</b>	<b>E-49</b>
Purpose . . . . .	E-49
Functional Block Diagram . . . . .	E-49
Pulse Catch Timing Parameters . . . . .	E-49
Setup for Mode 50 . . . . .	E-50
X Input Configuration . . . . .	E-50
Pulse Catch Program Example . . . . .	E-51
<b>Mode 60: Discrete Inputs with Filter . . . . .</b>	<b>E-52</b>
Purpose . . . . .	E-52
Functional Block Diagram . . . . .	E-52
Input Filter Timing Parameters . . . . .	E-52
Setup for Mode 60 . . . . .	E-53
X Input Configuration . . . . .	E-53
Filtered Inputs Program Example . . . . .	E-54

## Appendix F: PLC Memory

<b>DL05 PLC Memory . . . . .</b>	<b>F-2</b>
Non-volatile V-memory in the DL05 . . . . .	F-3

## Appendix G: ASCII Table

ASCII Table .....G-2

## Appendix H: Product Weights

Product Weight Table .....H-2

## Appendix I: Numbering Systems

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

## Appendix J: European Union Directives (CE)

European Union (EU) Directives .....J-2  
    Member Countries .....J-2  
    Applicable Directives .....J-2  
    Compliance .....J-2  
    General Safety .....J-3  
    Special Installation Manual .....J-4  
    Other Sources of Information .....J-4  
Basic EMC Installation Guidelines .....J-5  
    Enclosures .....J-5  
    Electrostatic Discharge (ESD) .....J-5  
    AC Mains Filters .....J-6



Suppression and Fusing .....	J-6
Internal Enclosure Grounding .....	J-6
Equi-potential Grounding .....	J-7
Communications and Shielded Cables .....	J-7
Analog and RS232 Cables .....	J-8
Multidrop Cables .....	J-8
Shielded Cables within Enclosures .....	J-9
Analog Modules and RF Interference .....	J-9
Network Isolation .....	J-9
DC Powered Versions .....	J-9
Items Specific to the DL05 .....	J-10

## Appendix K: Introduction to Serial Communications

<b>Introduction to Serial Communications .....</b>	<b>.K-2</b>
Wiring Standards .....	K-2
Communications Protocols .....	K-3
DL05 Port Specifications .....	K-4
DL05 Port Pinouts .....	K-4
Port Setup Using <i>DirectSOFT 5</i> or Ladder Logic Instructions .....	K-5
Port 2 Setup for RLL Using K-Sequence, <i>DirectNET</i> or Modbus RTU .....	K-6
Port 2 Setup for RLL Using ASCII .....	K-7
K-Sequence Communications .....	K-9
<i>DirectNET</i> Communications .....	K-9
Step 1: Identify Master Port # and Slave # .....	K-9
Step 2: Load Number of Bytes to Transfer .....	K-9
Step 3: Specify Master Memory Area .....	K-10
Step 4: Specify Slave Memory Area .....	K-11
Communications from a Ladder Program .....	K-12
Multiple Read and Write Interlocks .....	K-12
Modbus RTU Communications .....	K-13

## Index

# GETTING STARTED

---



# CHAPTER 1

## In This Chapter:

Introduction .....	1-2
Conventions Used .....	1-3
DL05 Micro PLC Components .....	1-4
Programming Methods .....	1-4
I/O Selection Quick Chart .....	1-5
Quick Start for PLC Checkout and Programming .....	1-6
Steps to Designing a Successful System .....	1-10
Questions and Answers about DL05 Micro PLCs .....	1-12

# 1 Introduction

## The Purpose of this Manual

Thank you for purchasing a DL05 Micro PLC. This manual shows you how to install, program, and maintain all the Micro PLCs in the DL05 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 DL05 PLCs, and for the PLC programmer. If you understand PLC systems our manuals will provide all the information you need to get and keep your system up and running.

## Where to Begin

If you already understand the DL05 Micro PLC please read Chapter 2, “Installation, Wiring, and Specifications”, and proceed on to other chapters as needed. Be sure to keep this manual handy for reference when you run into questions. If you are a new DL05 customer, we suggest you read this manual completely so you can understand the wide variety of features in the DL05 family of products. We believe you will be pleasantly surprised with how much you can accomplish with **AutomationDirect** products.

## Supplemental Manuals

The *DO-OPTIONS-M* manual will be most helpful to select and use any of the optional modules that are available for the DL05 PLC which includes the analog I/O modules. If you have purchased operator interfaces or *DirectSOFT* programming software you will need to supplement this manual with the manuals that are written for these products.

## Technical Support

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

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

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

- **Internet** – the address of our website is: <http://www.automationdirect.com>

If you still need assistance, please call us at 770-844-4200. Our technical support team will be available to work with you in answering your questions. They are available Monday through Friday from 9:00 A.M. to 6:00 P.M. Eastern Standard Time.

## Conventions Used



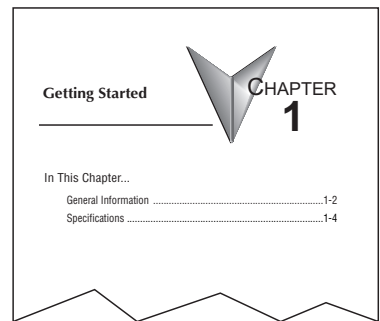
When you see the “notepad” icon in the left-hand margin, the paragraph to its immediate right will be a **special note**. Notes represent information that may make your work quicker or more efficient. The word **NOTE:** in boldface will mark the beginning of the text.



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.

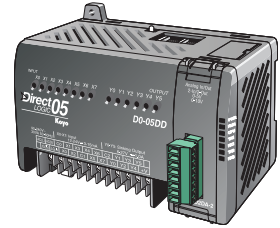
### Key Topics for Each Chapter

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



# 1 DL05 Micro PLC Components

The DL05 Micro PLC family is a versatile product line that provides a wide variety of features in a very compact footprint. The PLCs are small, yet offer many features usually found only in larger, more expensive systems. These include a removable connector, and two RS-232C communication ports.



## The DL05 Micro PLC Family

The DL05 Micro PLC family includes eight different versions. All have the same appearance and CPU performance. The CPU offers the same instruction set as our popular DL240 CPU, plus several more instructions specifically designed for machine control applications. All DL05 PLCs have two RS-232C communications ports. Units with DC inputs have selectable high-speed input features on three input points. Units with DC outputs offer selectable pulse output capability on the first and second output points. All DL05 Micro PLCs offer a large amount of program memory, a substantial instruction set and advanced diagnostics. Details of these features and more are covered in Chapter 3, CPU Specifications and Operation. The eight types of DL05 Micro PLCs provide a variety of Input/Output choices, listed in the following table.

DL05 Micro PLC Family					
DL05 Part Number	Discrete Input Type	Discrete Output Type	External Power	High-Speed Input	Pulse Output
DO-05AR	AC	Relay	95-240 VAC	No	No
DO-05DR	DC	Relay	95-240 VAC	Yes	No
DO-05AD	AC	DC	95-240 VAC	No	Yes
DO-05DD	DC	DC	95-240 VAC	Yes	Yes
DO-05AA	AC	AC	95-240 VAC	No	No
DO-05DA	DC	AC	95-240 VAC	Yes	No
DO-05DR-D	DC	Relay	12-24 VDC	Yes	No
DO-05DD-D	DC	DC	12-24 VDC	Yes	Yes

## DirectSOFT 5 Programming for Windows™

The DL05 Micro PLC can be programmed with one of the most advanced programming packages in the industry — *DirectSOFT 5*, 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.

*DirectSOFT 5* universally supports the *DirectLOGIC* CPU families. This means you can use the full version of *DirectSOFT 5* to program DL05, DL06, DL105, DL205, DL305, DL405 CPUs. The *DirectSOFT 5* Programming Software User Manual discusses the programming language in depth. *DirectSOFT* version 2.4 or later is needed to program the DL05.

## Handheld Programmer

All DL05 Micro PLCs have built-in programming ports for use with the handheld programmer (D2–HPP), the same programmer used with the DL06, 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 1.09 or later will program the DL05.



**NOTE:** Not all program instructions are available to use with the HHP, such as the DRUM instruction. Use *DirectSOFT 5* for these instructions.

## I/O Selection Quick Chart

The eight versions of the DL05 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 chart carefully to find the proper DL05 Micro PLC to interface to the field devices in your application.

I/O Selection Chart						
DL05 Part Number	INPUTS			OUTPUTS		
	I/O type/commons	Sink/Source	Voltage Ranges	I/O type/commons	Sink/Source	Voltage/ Current Ratings
D0–05AR	AC / 2	–	90 – 120 VAC	Relay / 2	Sink or Source	6 – 27VDC, 2A* 6 – 240 VAC, 2A *
D0–05DR	DC / 2	Sink or Source	12 – 24 VDC	Relay / 2	Sink or Source	6 – 27VDC, 2A* 6 – 240 VAC, 2A *
D0–05AD	AC / 2	–	90 – 120 VAC	DC / 1	Sink	6 – 27 VDC, 0.5A (Y0–Y2) 6 – 27 VDC, 1.0A (Y3–Y5)
D0–05DD	DC / 2	Sink or Source	12 – 24 VDC	DC / 1	Sink	6 – 27 VDC, 0.5A (Y0–Y2) 6 – 27 VDC, 1.0A (Y3–Y5)
D0–05AA	AC / 2	–	90 – 120 VAC	AC / 2	–	17 – 240 VAC, 47 – 63 Hz, 0.5A*
D0–05DA	DC / 2	Sink or Source	12 – 24 VDC	AC / 2	–	17 – 240 VAC, 47 – 63 Hz, 0.5A *
D0–05DR–D	DC / 2	Sink or Source	12 – 24 VDC	Relay / 2	Sink or Source	6 – 27 VDC, 2A 6 – 240 VAC, 2A *
D0–05DD–D	DC / 2	Sink or Source	12 – 24 VDC	DC / 1	Sink	6 – 27 VDC, 0.5A (Y0–Y2) 6 – 27 VDC, 1.0A (Y3–Y5)

\* See Chapter 2 Specifications for your particular DL05 version.

# 1 Quick Start for PLC Checkout and Programming

This example is not intended to tell you everything you need to start-up your system, warnings and helpful tips are in the rest of the manual. It is only intended to give you a general picture of what you will need to do to get your system powered-up.

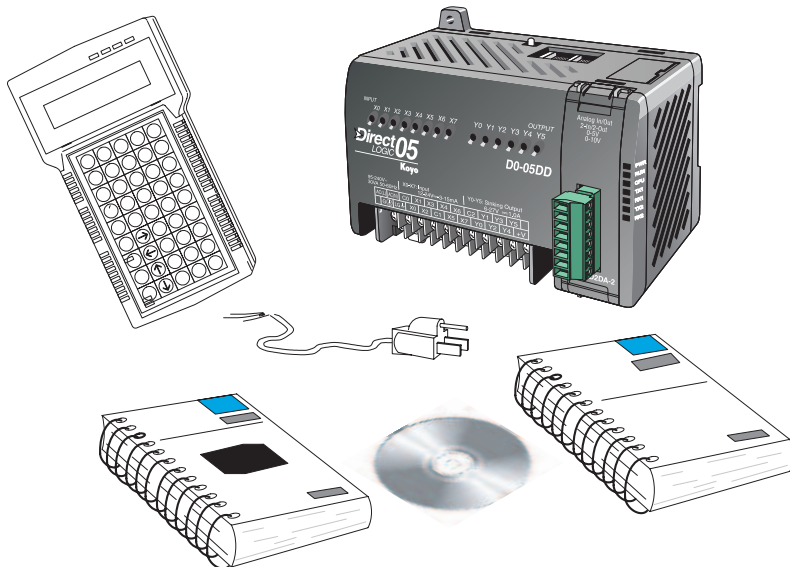
## Step 1: Unpack the DL05 Equipment

Unpack the DL05 and gather the parts necessary to build this demonstration system. The recommended components are:

- DL05 Micro PLC
- AC power cord or DC power supply
- Toggle switches or simulator module, F0-08SIM(see Step 2 on next page).
- Hook-up wire, 16-22 AWG
- DL05 User Manual (this manual)
- A small screwdriver, 5/8" flat or #1 Philips type

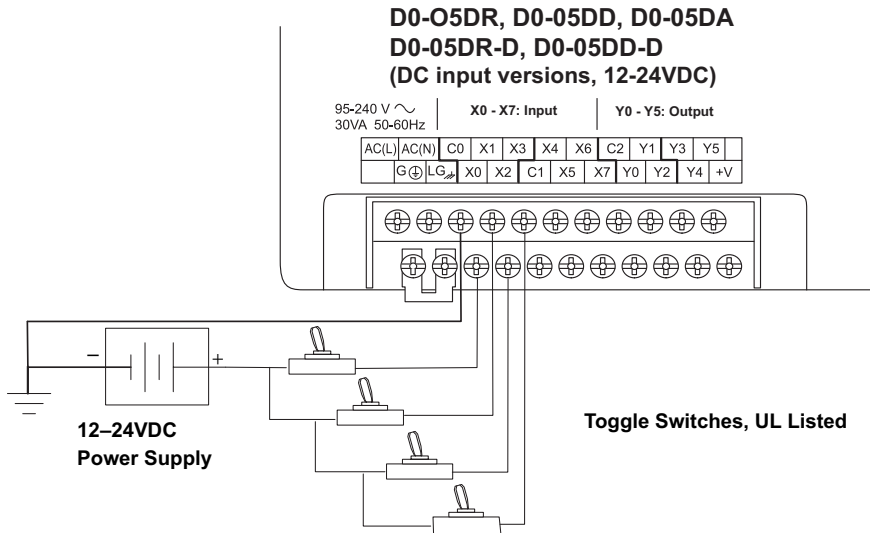
You will need at least one of the following programming options:

- *DirectSOFT 5* Programming Software, *DirectSOFT 5* Manual, and a programming cable (connects the DL05 to a personal computer), or
- D2-HPP Handheld Programmer (comes with programming cable), and the Handheld Programmer Manual

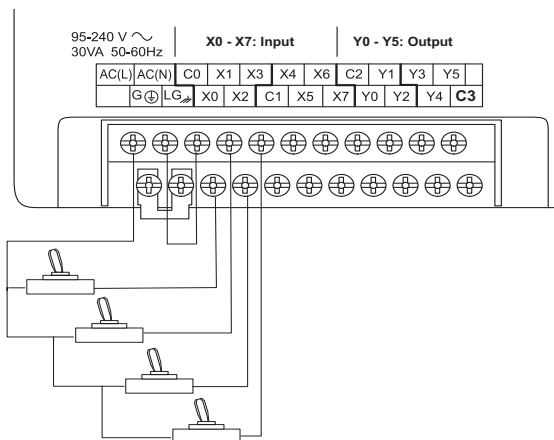


## Step 2: Connect Switches to Input Terminals

To finish this quick-start exercise or study other examples in this manual, you'll need to either connect some input switches as shown below or install the F0-08SIM, simulator module, which needs no wiring, into the option slot. If you have DC inputs you will need to use the FA-24PS (24VDC) or another external 12-24VDC power supply. Be sure to follow the instructions in the accompanying *WARNING* note.



### D0-05AR, D0-05AD, D0-05AA (AC input versions, 120V AC only)



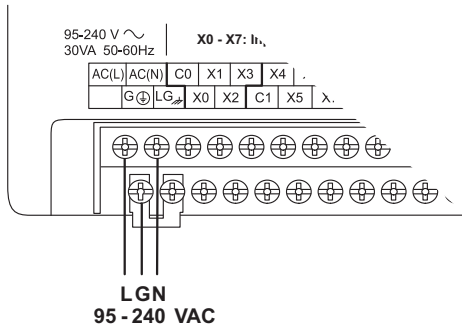
**WARNING: DO NOT** wire the toggle switches as shown to 240VAC-powered units. The discrete inputs will only accept 120VAC nominal. Also, remove power and unplug the DL05 when wiring the switches. Only use UL-approved switches rated for at least 250VAC, 1A for AC inputs. Firmly mount the switches before using.



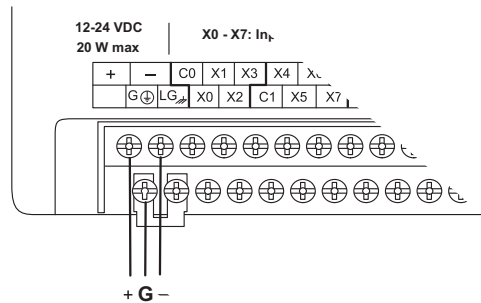
### Step 3: Connect the Power Wiring

Connect the power input wiring for the DL05. 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.

#### 110/220 VAC Power Input

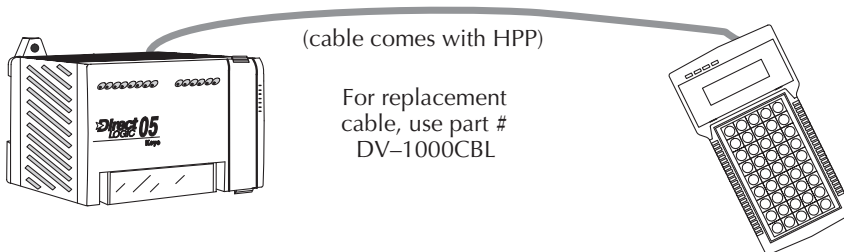
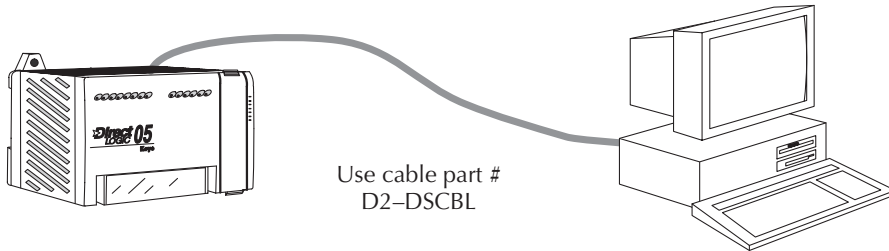


#### 12/24 VDC Power Input



### Step 4: Connect the Programming Device

Most programmers will use *DirectSOFT 5* programming software, installed on a personal computer. Or, you may need the portability of the Handheld Programmer. Both devices will connect to COM port 1 of the DL05 via the appropriate cable.



## Step 5: Switch on the System Power

Apply power to the system and ensure the PWR indicator on the DL05 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

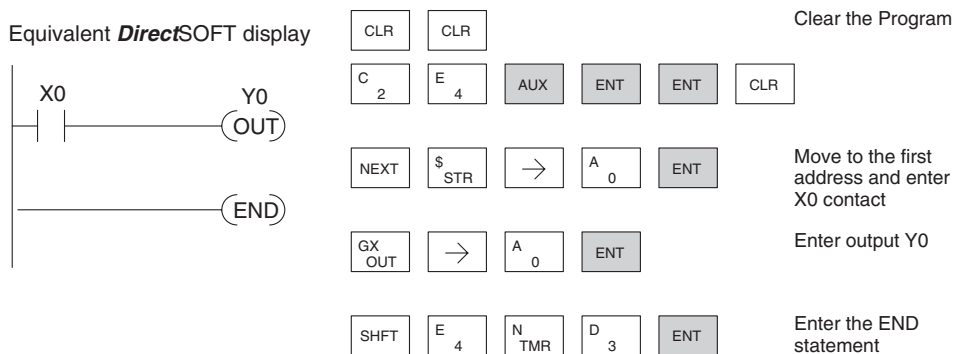
It's a good precaution to always clear the system memory (scratchpad memory) on a new DL05. There are two ways to clear the system memory:

- In *DirectSOFT 5*, select the PLC menu, then Setup, then Initialize Scratchpad. For additional information, see the *DirectSOFT 5 Manual*.
- 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, *DirectSOFT 5* programmers need to refer to the Quick Start Tutorial in the *DirectSOFT 5 Manual*. There you will learn how to establish a communications link with the DL05 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 DL05 should be off). If the RUN LED is on, use the MODE key on the Handheld Programmer to put the PLC in Program Mode. Enter the following keystrokes on the Handheld Programmer.



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

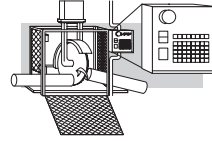
The RUN indicator on the PLC will illuminate indicating the CPU has entered the Run mode. If not, repeat this step, ensuring the program is entered properly or refer to the troubleshooting guide in chapter 9.

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.

# 1 Steps to Designing a Successful System

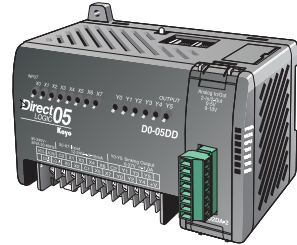
## Step 1: Review the Installation Guidelines

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 DL wiring guidelines for the various versions of the DL05 PLC.



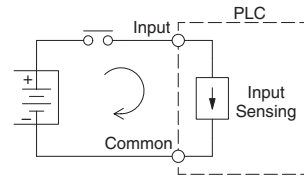
## Step 2: Understand the PLC Setup Procedures

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



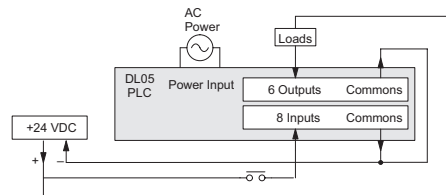
## Step 3: Review the I/O Selection Criteria

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.



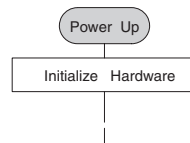
## Step 4: Choose a System Wiring Strategy

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 DL05 system processes information. This involves not only program execution steps, but also involves the various modes of operation and memory layout characteristics.

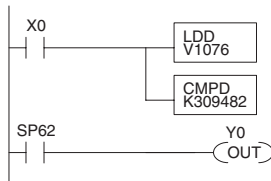


## Step 6: Review the Programming Concepts

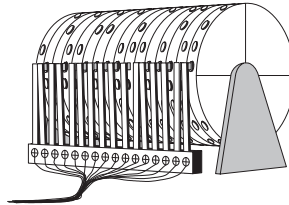
The DL05 PLC instruction set provides for three main approaches to solving the application program, depicted in the figure below.

- 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 RLL<sup>PLUS</sup>) 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.

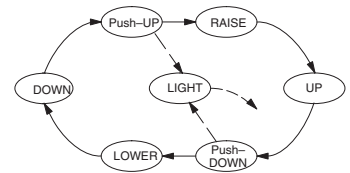
**Standard RLL Programming**  
(see Chapter 5)



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



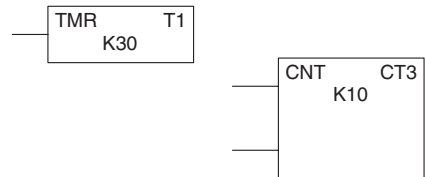
**Stage Programming**  
(see Chapter 7)



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

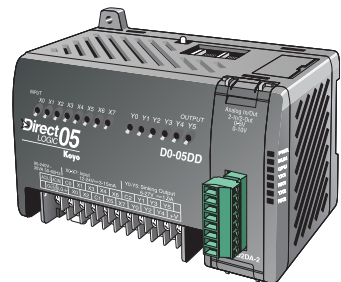
## Step 7: Choose the Instructions

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 DL05 Micro PLC has many built-in features such as error codes that can help you quickly identify problems.



## Questions and Answers about DL05 Micro PLCs

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

A. The instruction set is very close to our popular DL240 CPU. However, there are significant additions, such as the drum instruction, networking, PID control and High-Speed I/O capabilities.

**Q. Do I have to buy the full *DirectSOFT 5* programming package to program the DL05?**

A. No, *DirectSOFT 5* programming software is available for programming *DirectLOGIC* PLCs for no additional charge; however, it will only allow 100 maximum words to be programmed. Go to [AutomationDirect.com](http://AutomationDirect.com) for more information.

**Q. Is the DL05 expandable?**

A. No, the DL05 series are stand-alone PLCs with one slot for the installation of an available option module. They do not have expansion bases, such as our DL205 system which has expansion bases, yet are very compact and affordable.

**Q. Does the DL05 have motion control capability?**

A. Yes. The units with DC I/O have selectable high-speed input features on three inputs. There is also an optional High-Speed Counter I/O module available with special utility software. Either can accept pulse-type input signals for high-speed counting or timing applications and provide high-speed pulse-type output signals for stepper/servo motor control, monitoring, alarm or other discrete control functions. Three types of motion profiles are available, which are explained in Chapter 3.

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

A. The DL05 contains a non-removable FLASH memory for program storage, which may be written and erased thousands of times. You may transfer programs to/from the DL05 using *DirectSOFT 5* on a PC, or the HPP (which does support a removable EEPROM). There is an optional CMOS RAM memory cartridge (MC) available (See Chapter 10).

**Q. Does the DL05 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 DL05 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 DL05 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 DL05?**

**A. Port 1:** The port is RS-232C, fixed at 9600 baud, and uses the proprietary K-sequence protocol. The DL05 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 or Optimization Operator interface panels
- *DirectSOFT 5* (running on a personal computer)
- D2-HPP handheld programmer
- Other devices which communicate via K-sequence protocol should work with the DL05 Micro PLC. Contact the vendor for details.

**A. Port 2:** The port is RS-232C, 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 and non-sequence/print protocols.

**Q. Can the DL05 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

---



# CHAPTER 2

## In This Chapter:

Safety Guidelines .....	2-2
Orientation to DL05 Front Panel .....	2-5
Mounting Guidelines .....	2-7
Wiring Guidelines .....	2-11
System Wiring Strategies .....	2-14
Wiring Diagrams and Specifications .....	2-32
Glossary of Specification Terms .....	2-48

# Safety Guidelines

2



---

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

---



**WARNING: Providing a safe operating environment for personnel and equipment is your responsibility and should be your primary goal during system planning and installation. Automation systems can fail and may result in situations that can cause serious injury to personnel or damage to equipment. Do not rely on the automation system alone to provide a safe operating environment. You should use external electromechanical 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.

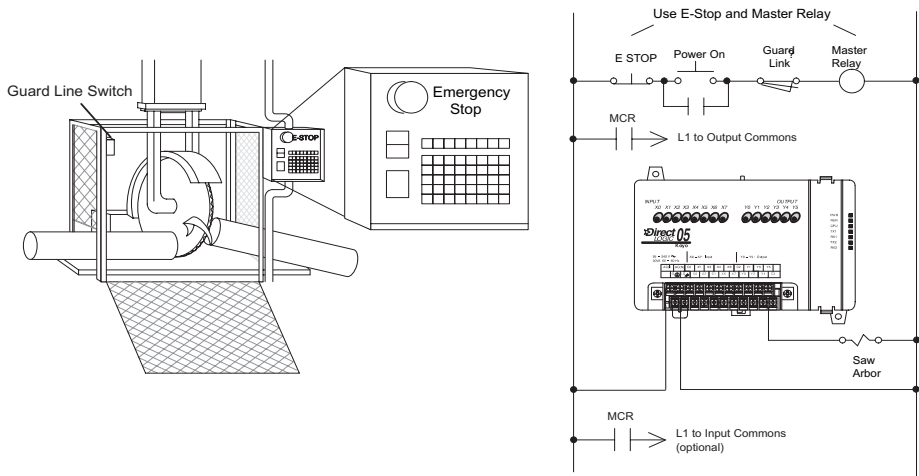
- Emergency stop switch for disconnecting system power
- Mechanical disconnect for output module power
- Orderly system shutdown sequence in the PLC control program

## Emergency Stops

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

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

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



### Emergency Power Disconnect

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

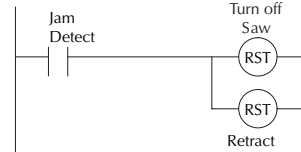
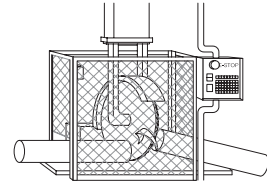
After an emergency shutdown or any other type of power interruption, there may be requirements that must be met before the PLC control program can be restarted. For example, there may be specific register values that must be established (or maintained from the state prior to the shutdown) before operations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

### Orderly System Shutdown

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

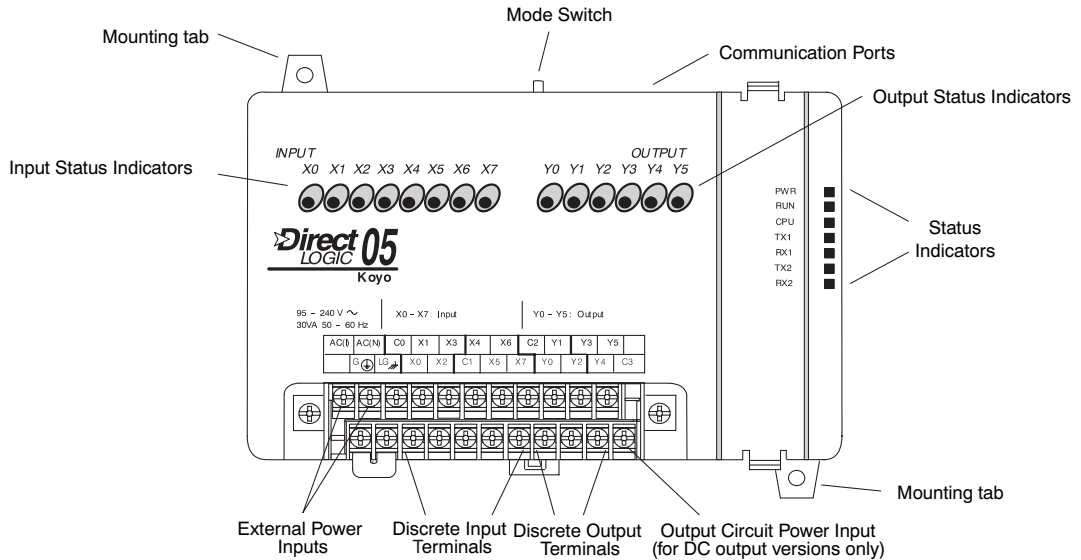


**WARNING: The control program must not be the only form of protection for any problems that may result in a risk of personal injury or equipment damage.**



## Orientation to DL05 Front Panel

Most connections, indicators, and labels on the DL05 Micro PLCs are located on its front panel. The communication ports are located on the top side of the PLC. Please refer to the drawing below.



The upper section of the connector accepts external power connections on the two left-most terminals. From left to right, the next five terminals are one of the input commons (C0) and input connections X1, X3, X4, and X6. The remaining four connections are an output common (C2) and output terminals Y1, Y3, and Y5.

The lower section of the connector has the chassis ground (G) and the logic ground (LG) on the two left-most terminals. The next two terminals are for the inputs X0 and X2. Next is the other input common (C1) followed by inputs X5 and X7. The last four terminals are for outputs Y0, Y2, Y4, and the second output common (C3). On DC output units, the end terminal on the right accepts power for the output stage.

An option slot is located on the right end of the PLC. This is where the simulator module can be installed for testing. Option module descriptions available for the DL05 can be found either in the DL05/DL06 Options Modules User Manual, D0-OPTIONS-M, in our catalog or on our website.



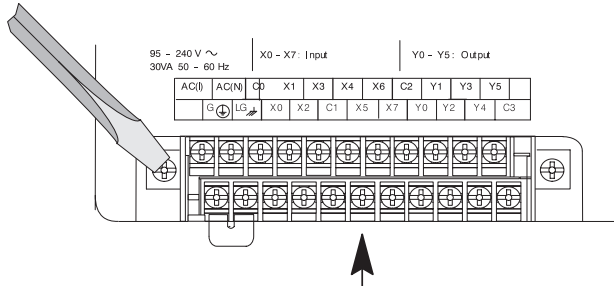
**WARNING:** For some applications, field device power may still be present on the terminal block even though the Micro PLC is turned off. To minimize the risk of electrical shock, check all field device power before you expose or remove either connector wire from under a terminal block, or two 18 AWG wires (one on each side of the screw).

### Connector Removal

All of the terminals for the DL05 are contained on one connector block. In some instances, it may be desirable to remove the connector block for easy wiring. The connector is designed for easy removal with just a small screwdriver. The drawing below shows the procedure for removal at one end.

#### Connector Removal

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 block connector on DL05 PLCs has regular screw terminals, which will accept either standard blade-type or #1 Philips screwdriver tips. Use No. 16 to 18 AWG solid/stranded wire. The maximum torque is 0.343Nm (3.036 inch-lbs).

Spare terminal block connectors and connector covers may be ordered by individual part numbers:

Spare Parts -Terminal Block and Cover		
Part Number	Qty Per Package	Description
F0-IOCON	2	DL05 I/O Terminal Block
D0-AAC-1	1 each	DL05 I/O Terminal Block, I/O Terminal Block Cover, and Option Slot Cover

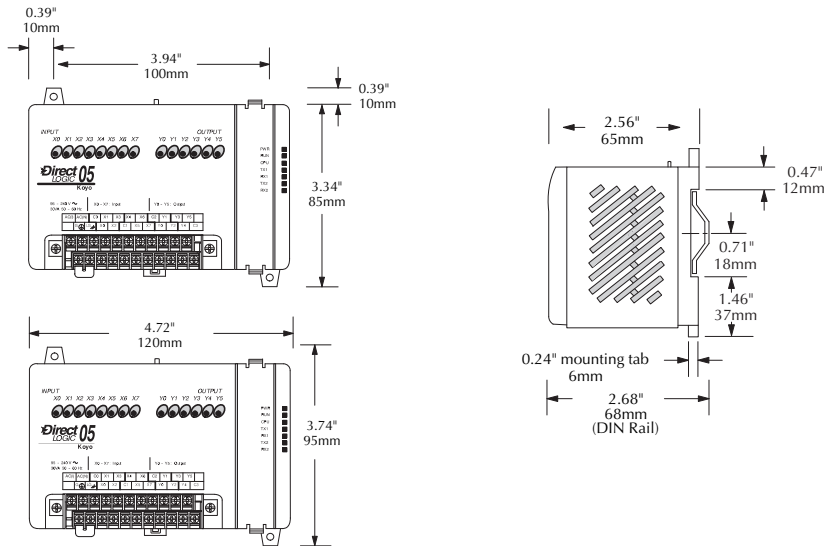
## Mounting Guidelines

In addition to the panel layout guidelines, other specifications can affect the definition and 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 DL05. Make sure you follow the installation guidelines to allow proper spacing from other components.



### Enclosures

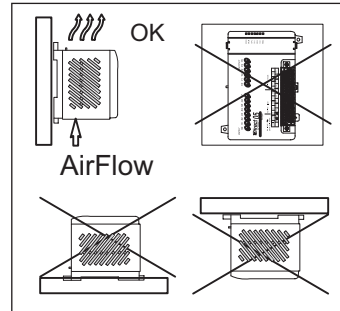
Your selection of a proper enclosure is important to ensure safe and proper operation of your DL05 system. Applications of DL05 systems vary and may require additional features. The minimum considerations for enclosures include:

- 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

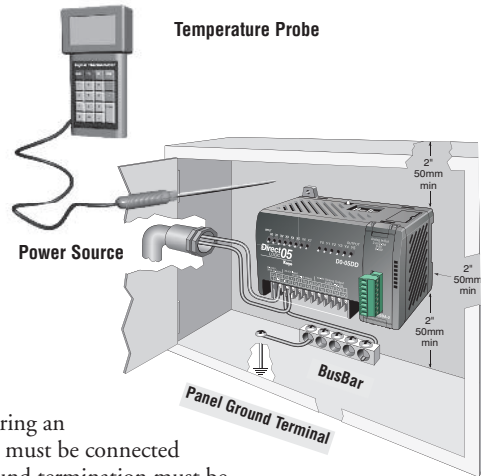
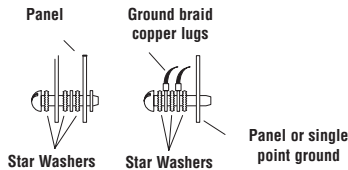
## Panel Layout & Clearances

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 DL05 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 2" (50mm) 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.
4. The ground terminal on the DL05 base must be connected to a single point ground. Use copper stranded wire to achieve a low impedance. Copper eye lugs should be crimped and soldered to the ends of the stranded wire to ensure good surface contact. Remove anodized finishes and use copper lugs and star washers at termination points.



**NOTE:** There is a minimum clearance requirement of 2" (51cm) between the panel door (or any devices mounted in the panel door) and the nearest DL05 component.



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 DL05. One side of all control and power circuits and the ground lead on flexible shielded cable must be properly connected to common ground reference. Methods which provide a good common ground reference, include:
  - 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 DL05 system, measures such as installing a cooling/heating source must be taken to get the ambient temperature within the range of specifications.

8. The DL05 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 DL05 PLCs from power surges and EMI/RFI noise. The Automation Powerline Filter, for use with 120 VAC and 240 VAC, 1-5 Amps (part number APF120N05), is an excellent choice, however, you can use a filter of your choice. These units install easily between the power source and the PLC.

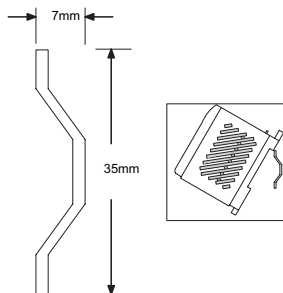


**NOTE:** If you are using other components in your system, make sure you refer to the appropriate manual to determine how those units can affect mounting dimensions.

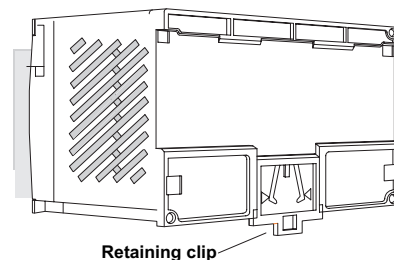
### Using DIN Rail Mounting Rails

DL05 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 is a small retaining clip. To secure the PLC to a DIN rail, place it onto the rail and gently push up on the clip to lock it onto the rail. To remove the PLC, pull down on the retaining clip, lift up on the PLC slightly, then pulling it away from the rail.

**Din Rail Dimensions**



**Din Rail Slot**  
Use 35mm x 7mm rail  
conforming to DIN EN 50022



**NOTE:** Refer to our catalog for a complete listing of **DINnector** connection systems.

### Environmental Specifications

The following table lists the environmental specifications that generally apply to DL05 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 DL05 PLC.

- \* 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.

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, FCC class A

### Agency Approvals

Some applications require agency approvals for particular components. The DL05 Micro PLC agency approvals are listed below:

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

### Marine Use

American Bureau of Shipping (ABS) certification requires flame-retarding insulation as per 4-8-3/5.3.6(a). ABS will accept Navy low smoke cables, cable qualified to NEC "Plenum rated" (fire resistant level 4), or other similar flammability resistant rated cables. Use cable specifications for your system that meet a recognized flame retardant standard (i.e. UL, IEEE, etc.), including evidence of cable test certification (i.e. tests certificate, UL file number, etc.).

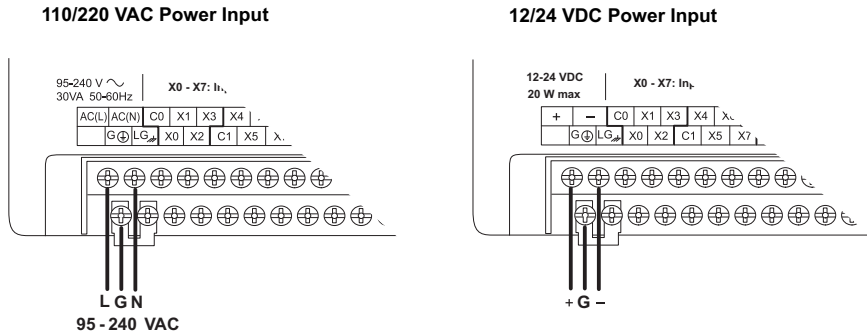


**NOTE:** Wiring needs to be "low smoke" per the above paragraph. Teflon coated wire is also recommended.



## Wiring Guidelines

Connect the power input wiring for the DL05. Observe all precautions stated earlier in this manual. Follow the guidelines in this chapter. 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 1A, such as AutomationDirect's AGC1, fast-acting fuse.

#### 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.

### 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 DL05 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 DL05. The recommended fuse size for the branch circuit to the DL05 is 1A, such as AutomationDirect's AGC1, fast-acting fuse.

### 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	DL05 VAC Powered Units	DL05 VDC Powered Units
<b>Input Voltage Range</b>	110/220 VAC (95–240 VAC)	12–24 VDC (10.8–26.4 VDC)
<b>Maximum Inrush Current</b>	13 A, 1ms (95–240 VAC) 15 A, 1ms (240–264 VAC)	10A < 1ms
<b>Maximum Power</b>	30 VA	20 W
<b>Voltage Withstand (dielectric)</b>	1 minute @ 1500 VAC between primary, secondary, field ground	
<b>Insulation Resistance</b>	> 10 MΩ at 500 VDC	



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



**NOTE:** It is possible to use an uninterruptible power supply (UPS); however, the output must be a sinusoidal waveform for the PLC to perform properly.

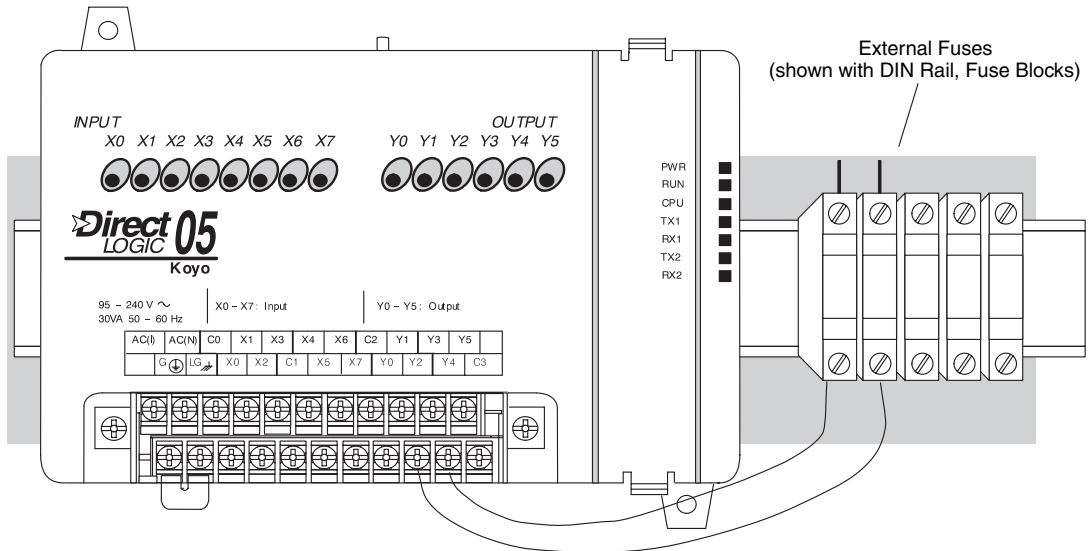
### Planning the Wiring Routes

The following guidelines provide general information on how to wire the I/O connections to DL05 Micro PLCs. For specific information on wiring a particular PLC refer to the corresponding specification sheet further in this chapter.

1. Each terminal connection of the DL05 PLC can accept one 16 AWG wire or two 18 AWG size wires. Do not exceed this recommended capacity.
2. Always use a continuous length of wire. Do not splice wires to attain a needed length.
3. Use the shortest possible wire length.
4. Use wire trays for routing where possible.
5. Avoid running wires near high energy wiring.
6. Avoid running input wiring close to output wiring where possible.
7. To minimize voltage drops when wires must run a long distance, consider using multiple wires for the return line.
8. Avoid running DC wiring in close proximity to AC wiring where possible.
9. Avoid creating sharp bends in the wires.
10. 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 DL05 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 tables 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 DL05 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 reference addresses are typically assigned in groups of 8 or 16, depending on the number of points in an I/O group. For the DL05 the eight inputs use reference numbers X0 – X7. The six output points use references Y0 – Y5.

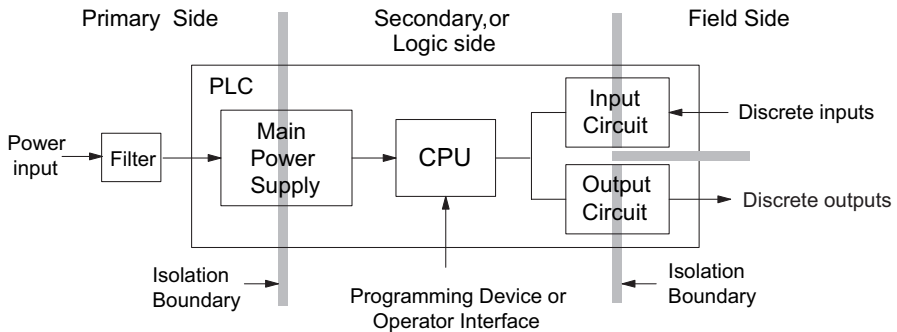
If an optional input module is installed in the option slot, the reference numbers are X100 – X107. A typical output module installed in the option slot will have references Y100 – Y107. See the DL05/06 Options Modules User Manual (D0-OPTIONS-M) for addressing other optional modules.

## System Wiring Strategies

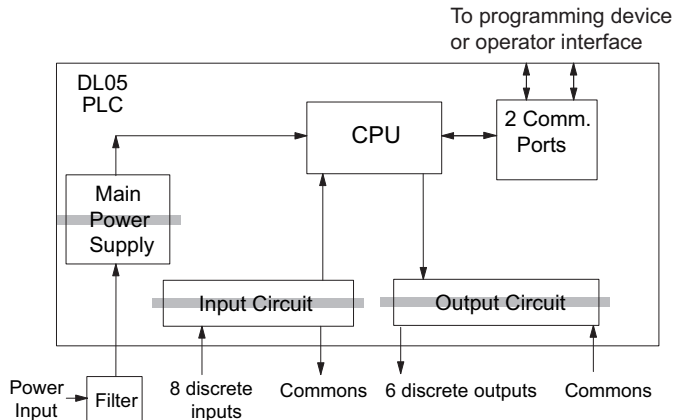
The DL05 Micro PLC is very flexible and will work in many different wiring configurations. By studying this section before actual installation, you can probably find the best wiring strategy for your application. This will help to lower system cost, wiring errors, and avoid safety problems.

### 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 which connect to the logic side of the PLC circuits.*

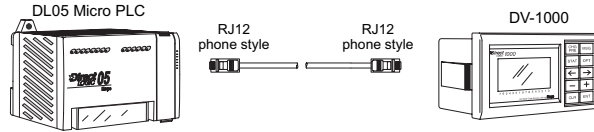


The next figure shows the internal layout of DL05 PLCs, as viewed from the front panel.

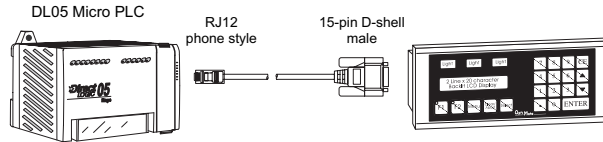


### Connecting Operator Interface Devices

Operator interfaces require data and power connections. Operator interfaces with a large CRT usually require separate AC power. However, some small operator interface devices, such as the DV-1000 Data Access Unit, may be powered directly from the DL05 Micro PLC. Connect the DV-1000 to communication port 1 on the DL05 Micro PLC with a DV-1000CBL. A single cable contains transmit/receive data wires and +5V power.

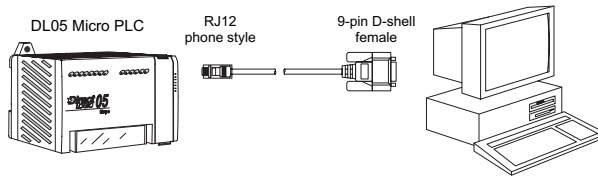


Operator interface panels require separate power and communications connections. Connect the DL05 with the proper cable and power supply for your application.

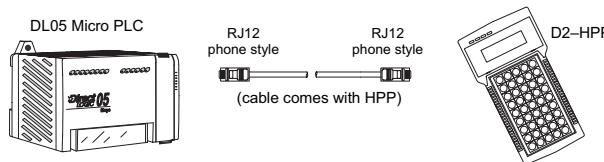


### Connecting Programming Devices

DL05 Micro PLCs are programmed using *DirectSOFT 5* programming software on a PC. Limited programming can be accomplished with a handheld programmer. The DL05 can be interfaced to the PC with either a serial cable shown below or a Ethernet cable (Ethernet requires either an H0-ECOM or H0-ECOM100 module installed in the option slot). See the *AutomationDirect* catalog to chose the cable or Ethernet card for whichever method will meet your system requirements.



The D2-HPP Handheld Programmer comes with a communications cable. For a replacement cable, order a DV-1000CBL.



## Sinking/Sourcing Concepts

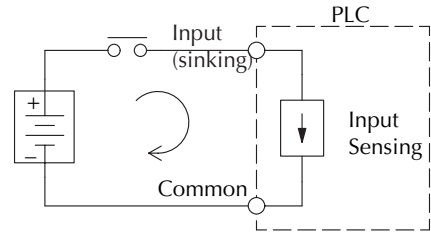
Before going further in our study of wiring strategies, we must have a solid understanding of “sinking” and “sourcing” concepts. Use of these terms occurs frequently in input or output circuit discussions. It is the goal of this section to make these concepts easy to understand, further ensuring your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

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

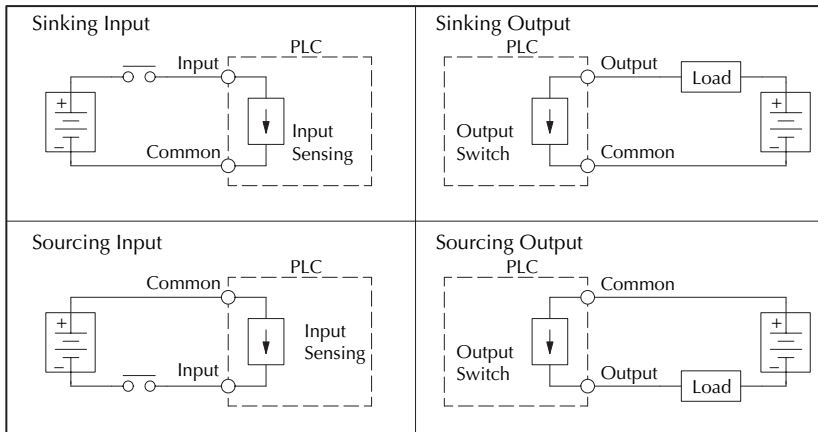
**Sourcing = Path to supply source (+)**

First you will notice that these are only associated with DC circuits and not AC, because of the reference to (+) and (-) polarities. Therefore, *sinking and sourcing terminology only applies to DC input and output circuits*. Input and output points that are 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 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.

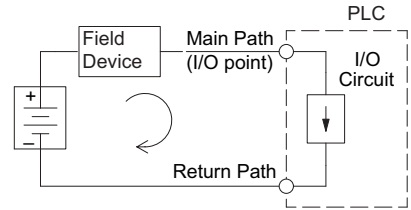


By applying the circuit principle above to the four possible combinations of input/output sinking/sourcing types, we have the four circuits as shown below. DL05 Micro PLCs provide all except the sourcing output I/O circuit types.

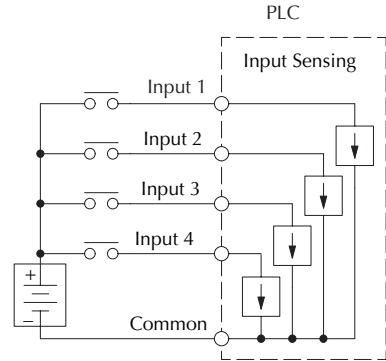


### I/O “Common” Terminal Concepts

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

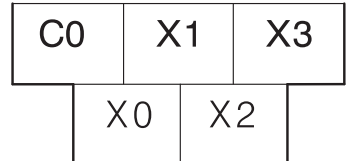


If we had unlimited space and budget for I/O terminals, then every I/O point could have two dedicated terminals just as the figure above shows. However, providing this level of flexibility is not practical or even necessary for most applications. So, most Input or Output point groups on PLCs share the 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 above, the current in the common path is 4 times any channel’s input current when all inputs are energized. This is especially important in output circuits, where heavier gauge wire is sometimes necessary on commons.

Most DL05 input and output circuits are grouped into banks that share a common return path. The best indication of I/O The I/O common grouping bar, labeled at the right, occurs in the section of wiring label below it. It indicates X0, X1, X2, and X3 share the common terminal located to the left of X1.



The following complete label shows two banks of four inputs and two banks of three outputs. One common is provided for each bank.

AC(L)	AC(N)	C0	X1	X3	X4	X6	C2	Y1	Y3	Y5	
	G ⊕	LG ⚡	X0	X2	C1	X5	X7	Y0	Y2	Y4	C3

The following label is for DC output versions. One common is provided for all of the outputs and the terminal on the bottom right accepts power for the output stage.

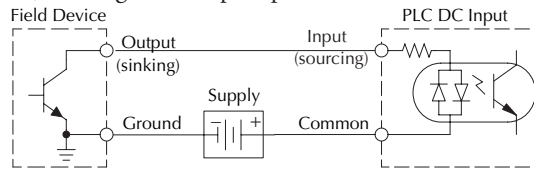
AC(L)	AC(N)	C0	X1	X3	X4	X6	C2	Y1	Y3	Y5	
	G ⊕	LG ⚡	X0	X2	C1	X5	X7	Y0	Y2	Y4	+V

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

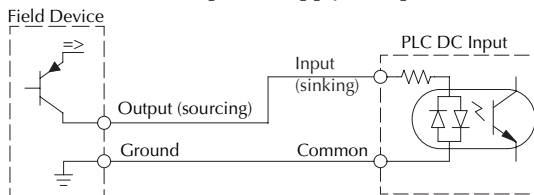
In the previous section on Sourcing and Sinking concepts, we explained that DC I/O circuits sometimes will only allow current to flow one way. This is also true for many of the field devices which have solid-state (transistor) interfaces. In other words, field devices can also be sourcing or sinking.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

### Solid State Input Sensors

The DL05'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 FA-24PS +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.

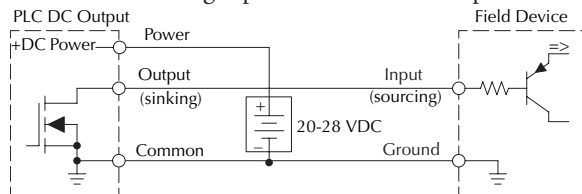


### Solid State Output Loads

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.

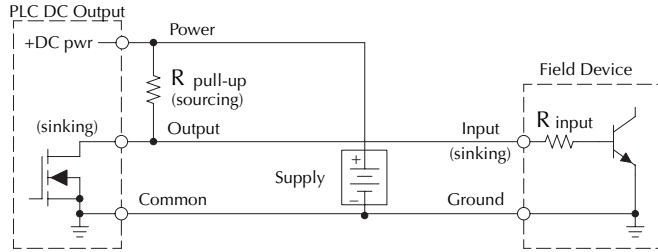
The DL05's DC outputs are sinking-only. This means that each DC output provides a path to ground when it is energized. Also, remember that all six outputs have the same electrical common, even though there are two common terminal screws. Finally, recall that the DC output circuit requires power (20 – 28 VDC) from an external power source.

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  $R_{pull-up}$  from the output to the DC output circuit power input.



**NOTE 1:** DO NOT attempt to drive a heavy load (>25 mA) with this pull-up method.

**NOTE 2:** Using the pull-up resistor to implement a sourcing output has the effect of inverting the output point logic. In other words, the field device input is energized when the PLC output is OFF, from a ladder logic point-of-view. Your ladder program must comprehend this and generate an inverted output. Or, you may choose to cancel the effect of the inversion elsewhere, such as in the field device.

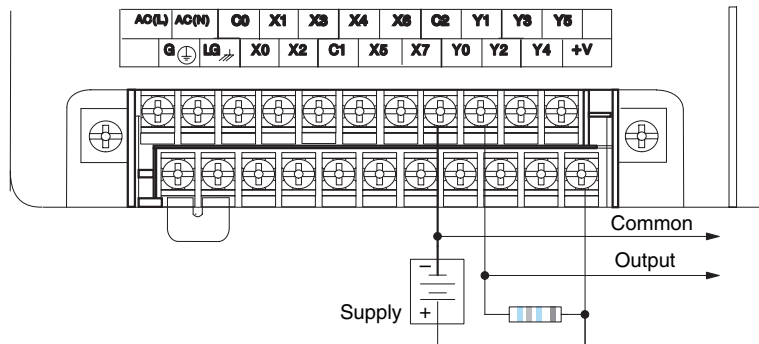
It is important to choose the correct value of  $R_{pull-up}$ . In order to do so, we need to know the nominal input current to the field device ( $I_{input}$ ) when the input is energized. If this value is not known, it can be calculated as shown (a typical value is 15 mA). Then use  $I_{input}$  and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

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

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

$P_{pull-up}$  (in watts), in order to size  $R_{pull-up}$  properly.

The drawing below shows the actual wiring of the DL05 Micro PLC to the supply and pull-up resistor.



## Relay Output Wiring Methods

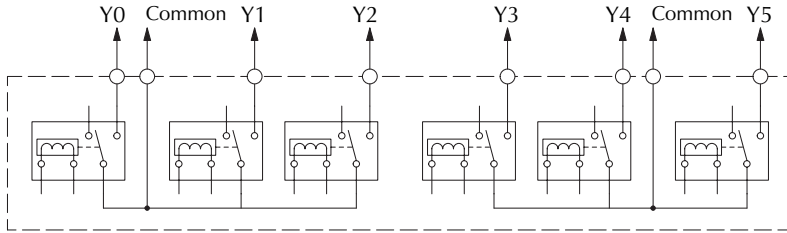
The D0–05AR and the D0–05DR models feature relay outputs. Relays are best for the following applications:

- Loads that require higher currents than the solid-state DL05 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)

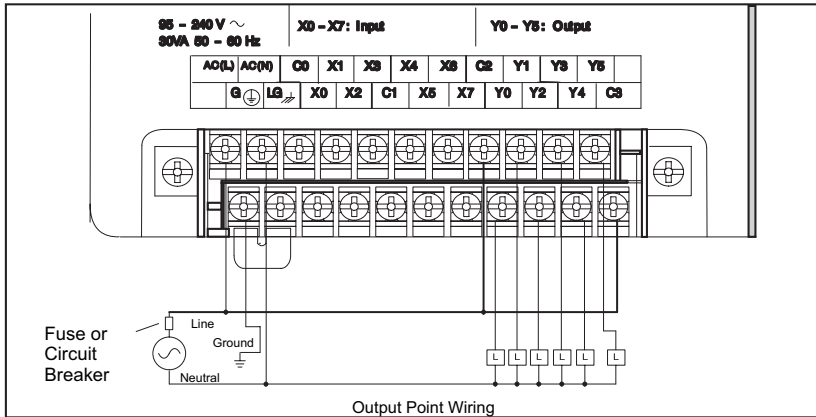
Some applications in which NOT to use relays:

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

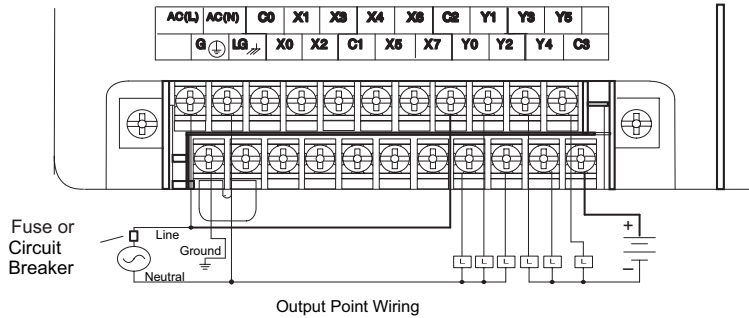
Assuming relays are right for your application, we're now ready to explore various ways to wire relay outputs to the loads. Note that there are six normally-open SPST relays available. They are organized with three relays per common. The figure below shows the relays and the internal wiring of the PLC. 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 DL05 PLC. In this example, all commons are connected together.



In the circuit on the following page, loads for Y0 – Y2 use the same AC power supply which powers the DL05 PLC. Loads for Y3 – Y5 use a separate DC supply. In this example, the commons are separated according to which supply powers the associated load.



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

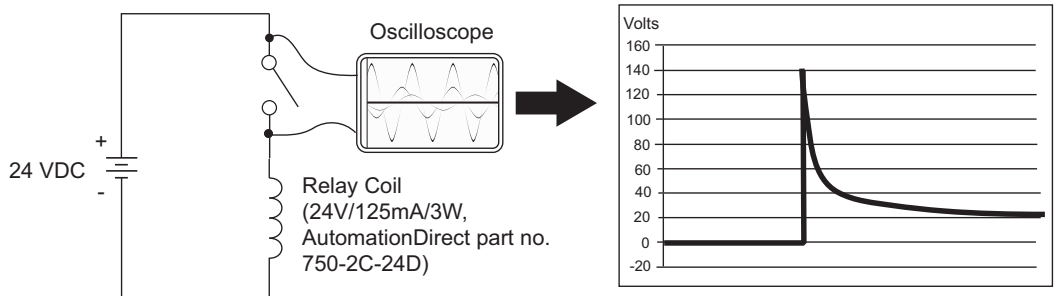
The following pages are intended to give a quick overview of the negative effects of transient voltages on a control system and provide some simple advice on how to effectively minimize them. The need for transient suppression is often not apparent to the newcomers in the automation world. Many mysterious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

#### What is a Transient Voltage and Why is it Bad?

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

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

#### Example: Circuit with no Suppression



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

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

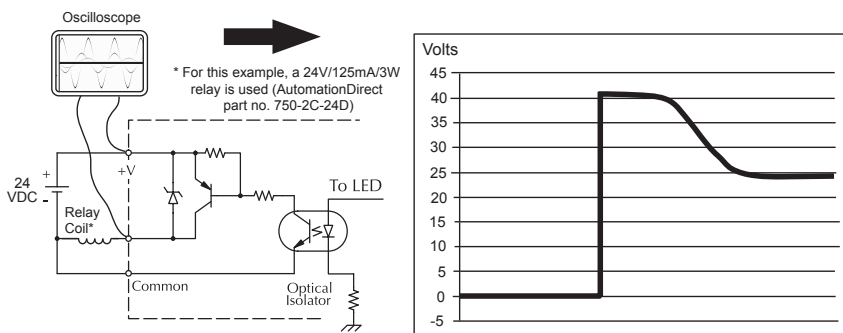
A very destructive side-effect of the arcing across relay contacts is the electromagnetic interference (EMI) it can cause. This occurs because the arcing causes a current surge, which releases RF energy. The entire length of wire between the relay contacts, the coil, and the power source carries the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

### PLC's Integrated Transient Suppressors

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

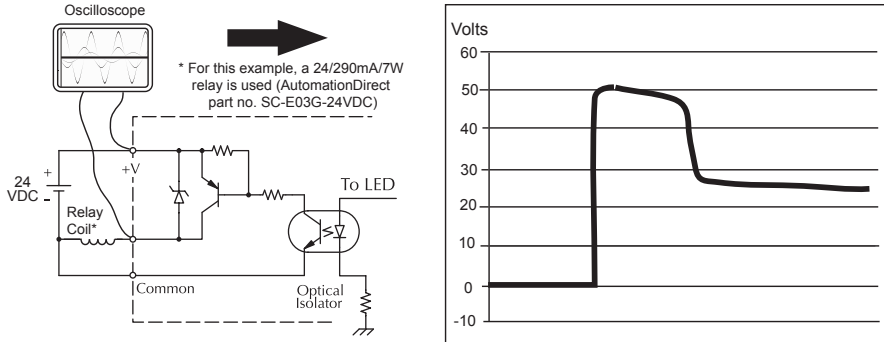
Here is another example using the same 24 V/125 mA/3 W relay used earlier. This example measures the PNP transistor output of a D0-06DD2 PLC, which incorporates an integrated Zener diode for transient suppression. Instead of the 140 V peak in the first example, the transient voltage here is limited to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

#### Example: Small Inductive Load with Only Integrated Suppression



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

**Example: Larger Inductive Load with Only Integrated Suppression**

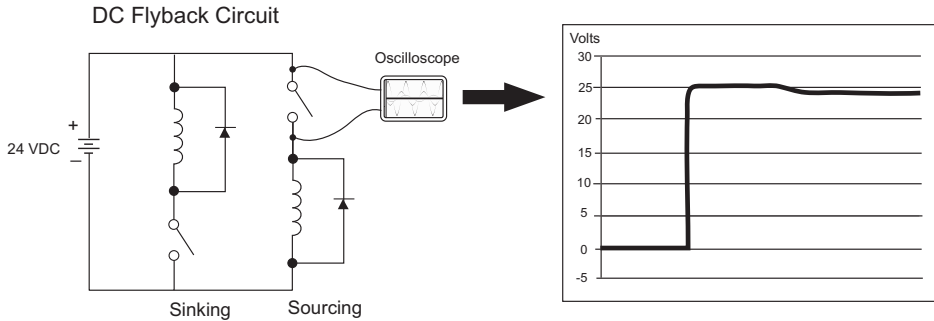


Additional transient suppression should be used in both these examples. If you are unable to measure the transients generated by the connected loads of your control system, using additional transient suppression on all inductive loads would be the safest practice.

**Types of Additional Transient Protection**

**DC Coils:**

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



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



**AD-ASMD-250**  
**Protection Diode Module**



**784-4C-SKT-1**  
**Relay Socket**



**DN-D10DR-A**  
**Diode Terminal Block**

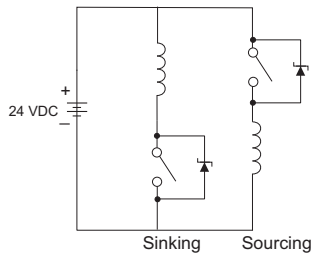
Two more common options for DC coils are Metal Oxide Varistors (MOV) or TVS diodes. These devices should be connected across the driver (PLC output) for best protection as shown below. The optimum voltage rating for the suppressor is the lowest rated voltage available that will NOT conduct at the supply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

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



**ZL-TSD8-24**  
**Transorb Module**

DC MOV or TVS Diode Circuit



**AC Coils:**

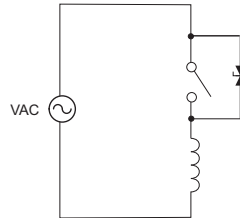
Two options for AC coils are MOVs or bi-directional TVS diodes. These devices are most effective at protecting the driver from a transient voltage when connected across the driver (PLC output) but are also commonly connected across the coil. The optimum voltage rating for the suppressor is the lowest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

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



**ZL-TSD8-120**  
Transorb Module

AC MOV or Bi-Directional Diode Circuit

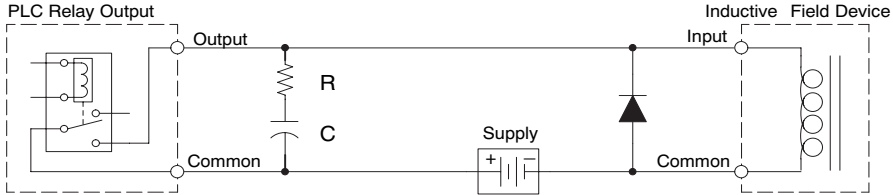


**NOTE:** Manufacturers of devices with coils frequently offer MOV or TVS diode suppressors as an add-on option which mount conveniently across the coil. Before using them, carefully check the suppressor's ratings. Just because the suppressor is made specifically for that part does not mean it will reduce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

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

## Prolonging Relay Contact Life

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



Adding external contact protection may extend relay life beyond the number of contact cycles listed in the specification tables for relay modules. High current inductive loads such as clutches, brakes, motors, direct-acting solenoid valves and motor starters will benefit the most from external contact protection.

When using an RC network, locate it close to the relay module output connector. To find the values for the RC snubber network, first determine the voltage across the contacts when open, and the current through them when closed. If the load supply is AC, then convert the current and voltage values to peak values.

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

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

C minimum = 0.001  $\mu\text{F}$ , the voltage rating of C must be  $\geq V$ , non-polarized

R minimum = 0.5  $\Omega$ , 1/2 W, tolerance is  $\pm 5\%$

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

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

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



Now, finding the values of R and C,:

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

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

$$x = 1 + \frac{50}{169.7} = 1.29 \quad R (\Omega) = \frac{169.7}{10 \times 0.707^{1.29}} = 26 \Omega, 1/2 \text{ W, } \pm 5\%$$

For inductive loads in DC circuits we recommend using a suppression diode as shown in the diagram. 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.




---

**WARNING: DO NOT use this circuit with an AC power supply.**

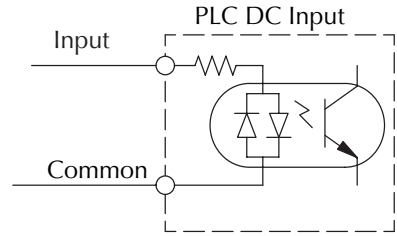
---

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, 3 A 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 will short-circuit the supply when the relay energizes.

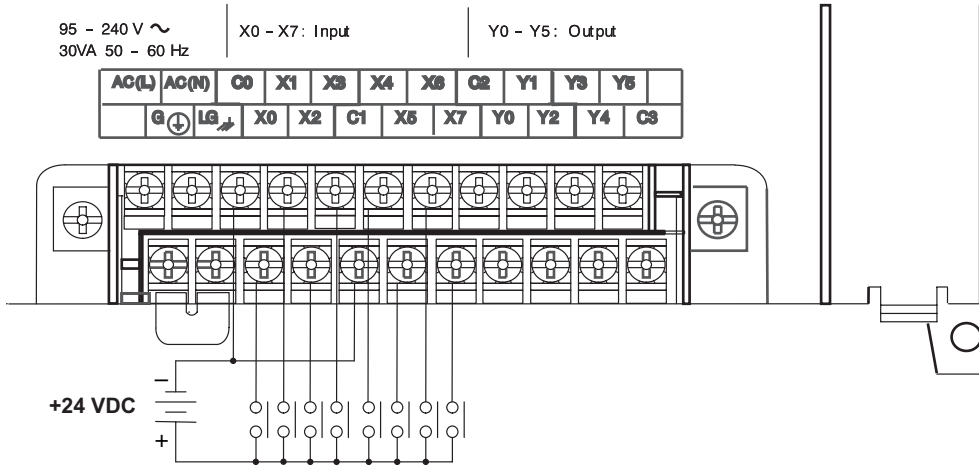
DO NOT use this circuit with an AC power supply.

### DC Input Wiring Methods

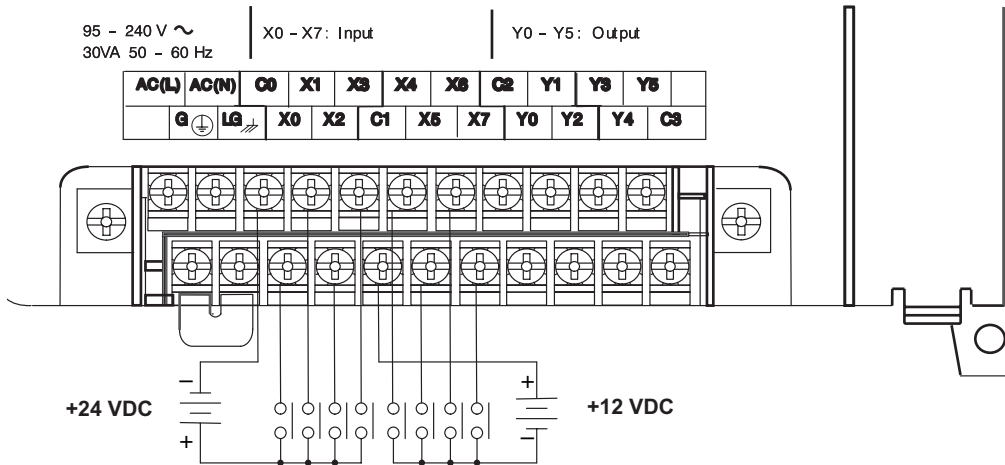
DL05 Micro PLCs with DC inputs are particularly flexible because they can be either sinking or sourcing. The dual diodes (shown to the right) allow 10.8 – 26.4 VDC. The target applications are +12 VDC and +24 VDC. You can actually wire half of the inputs as DC sinking and the other half as DC sourcing. 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 four inputs are sinking, and the last four are sourcing.

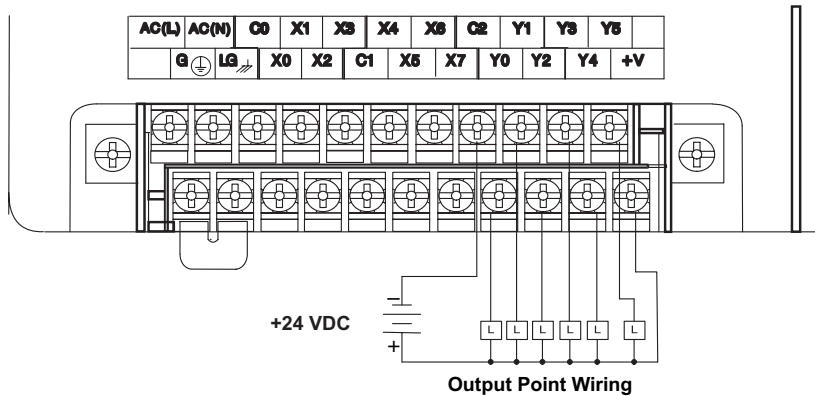


## DC Output Wiring Methods

DL05 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:

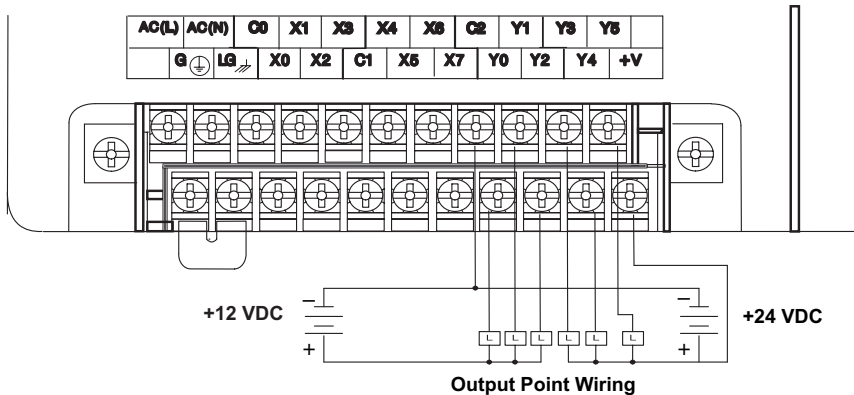
- There is only one electrical common for all six outputs. All six outputs belong to one bank.
- The output switches are current-sinking only. However, you can still use different DC voltages from one load to another.
- The output circuit inside the PLC requires external power. The supply (–) must be connected to a common terminal, and the supply (+) connects the right-most terminal on the upper connector.

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



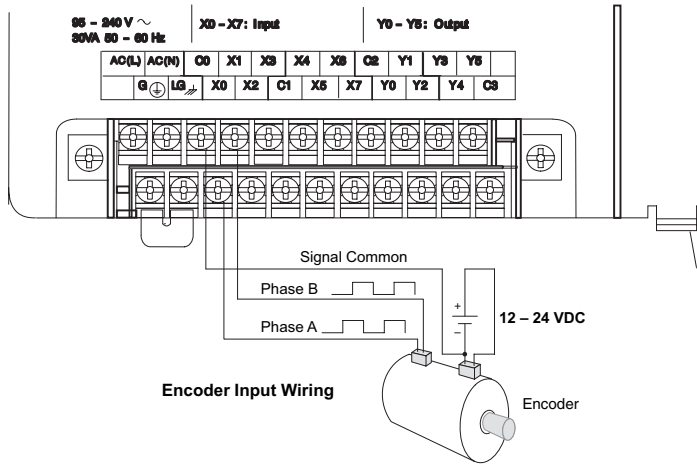
In the next example below, the outputs have “split” supplies. The first three outputs are using a +12 VDC supply, and the last three are using a +24 VDC supply. However, you can split the outputs among any number of supplies, as long as:

- all supply voltages are within the specified range
- all output points are wired as sinking
- all source (–) terminals are connected together

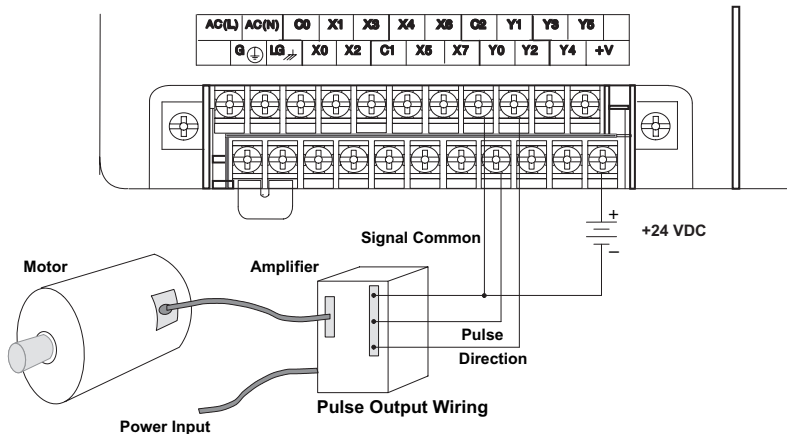


## High Speed I/O Wiring Methods

DL05 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 select 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 – X2. Properly configured, the DL05 can count quadrature pulses at up to 5 kHz.



DL05 versions with DC type output points can use the High Speed I/O Pulse Output feature. It can generate high-speed pulses up to 7 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 Appendix E on high-speed input and pulse output options.



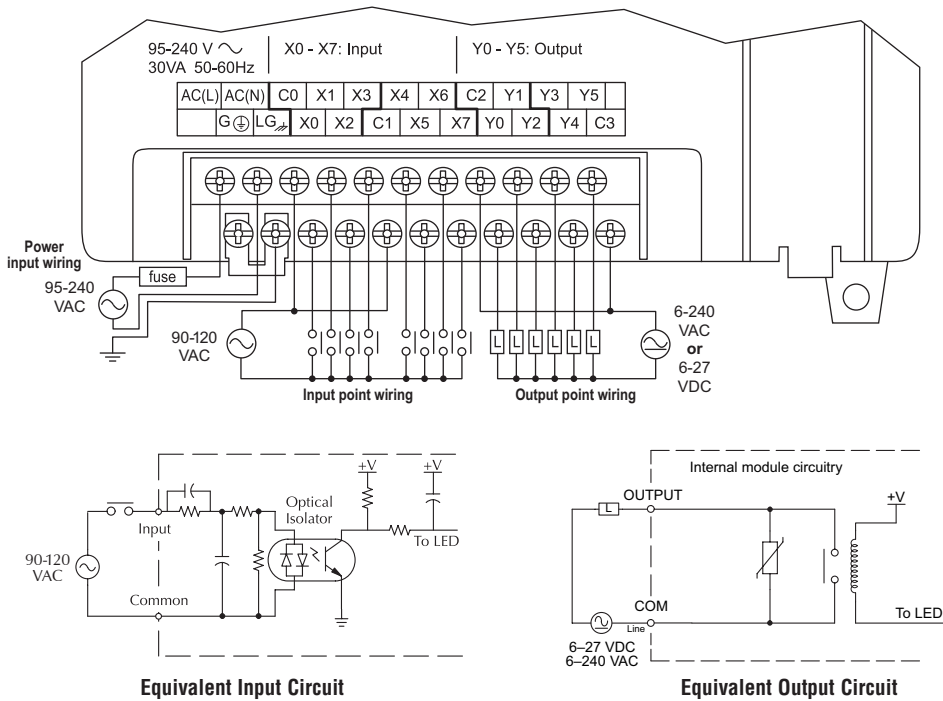
**This page is intentionally left blank.**

# Wiring Diagrams and Specifications

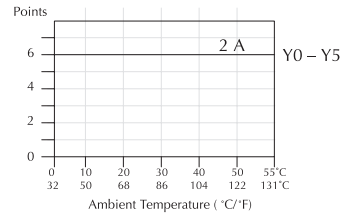
The remainder of this chapter dedicates two pages to each of the eight versions of DL05 Micro PLCs. Each section contains a basic wiring diagram, equivalent I/O circuits, and specification tables. Please refer to the section which describes the particular DL05 version used in your application.

## D0-05AR I/O Wiring Diagram

The D0-05AR Micro PLC features eight AC inputs and six 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 eight AC input channels use terminals in the middle of the connector. Inputs are organized into two banks of four. 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 input circuit shows one channel of a typical bank.



Derating Chart for Relay Outputs

The six relay output channels use terminals on the right side of the 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.

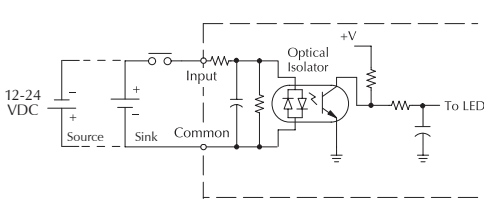
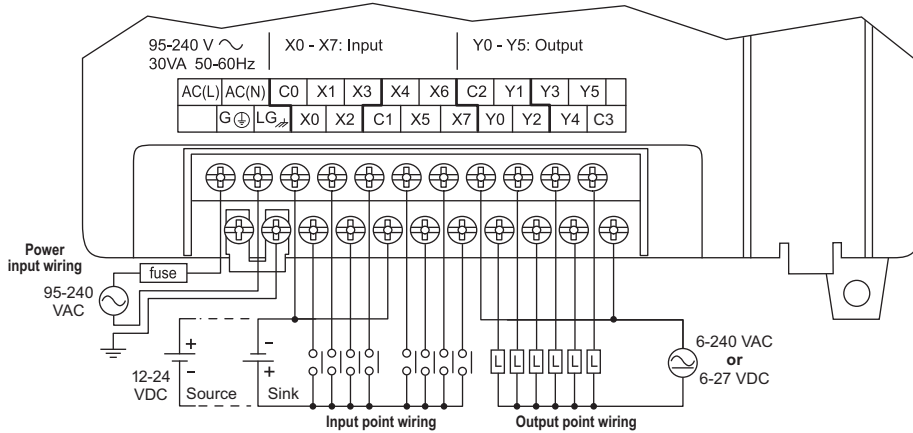
DO-05AR General Specifications	
External Power Requirements	95 – 240 VAC, 30 VA maximum,
Communication Port 1, 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), <b>DirectNET</b> (Slave), Modbus (Slave)
Communication Port 2, 9600 baud (default), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), <b>DirectNET</b> (Master/Slave), Modbus (Master/Slave) Non-sequence/print
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 16AWG or two 18AWG, 24AWG minimum

AC Input Specifications X0-X7	
Input Voltage Range (Min. - Max.)	80 – 132 VAC, 47 - 63 Hz
Operating Voltage Range	90 – 120 VAC, 47 -63 Hz
Input Current	8 mA @ 100 VAC at 50 Hz, 10 mA @ 100 VAC at 60 Hz
Max. Input Current	12 mA @ 132 VAC at 50 Hz, 15 mA @ 132 VAC at 60 Hz
Input Impedance	14KΩ @50 Hz, 12KΩ @60 Hz
ON Current/Voltage	>6 mA @ 75 VAC
OFF Current/Voltage	<2 mA @ 20 VAC
OFF to ON Response	< 40 ms
ON to OFF Response	< 40 ms
Status Indicators	Logic Side
Commons	4 channels/common x 2 banks

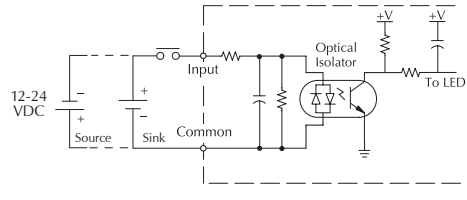
Relay Output Specifications Y0-Y5	
Output Voltage Range	(Min. – Max.) 5 – 264 VAC (47 -63 Hz), 5 – 30 VDC
Operating Voltage Range	6 – 240 VAC (47 -63 Hz), 6 – 27 VDC
Output Current	2A/point, 6A/ common
Max. leakage current	0.1 mA @264VAC
Smallest Recommended Load	5 mA @5 VDC
OFF to ON Response	< 15 ms
ON to OFF Response	< 10 ms
Status Indicators	Logic Side
Commons	3 channels/common x 2 banks
Fuses	None (external recommended)

### D0-05DR I/O Wiring Diagram

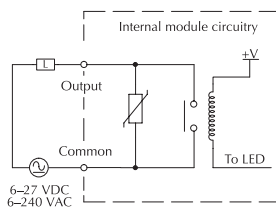
These micro PLCs feature eight DC inputs and six relay contact outputs. The following diagram shows a typical field wiring example. The AC external power connection uses four terminals at the left as shown.



Equivalent Circuit, High-speed Inputs (X0-X2)

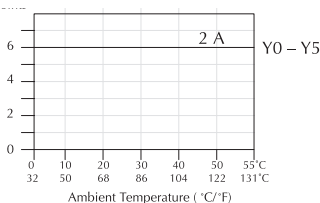


Equivalent Circuit, Standard Inputs (X3-X7)



Equivalent Output Circuit

The eight DC input channels use terminals in the middle of the connector. Inputs are organized into two 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. The equivalent circuit for standard inputs and the high-speed input circuit are shown above.



Derating Chart for Relay Outputs

The six output channels use terminals on the right side of the connector. Outputs are organized into two banks of three 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.



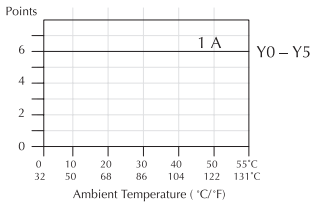
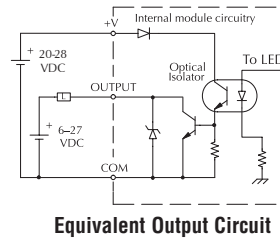
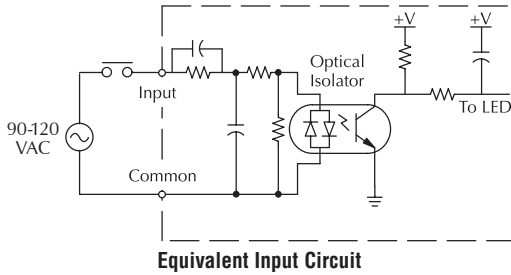
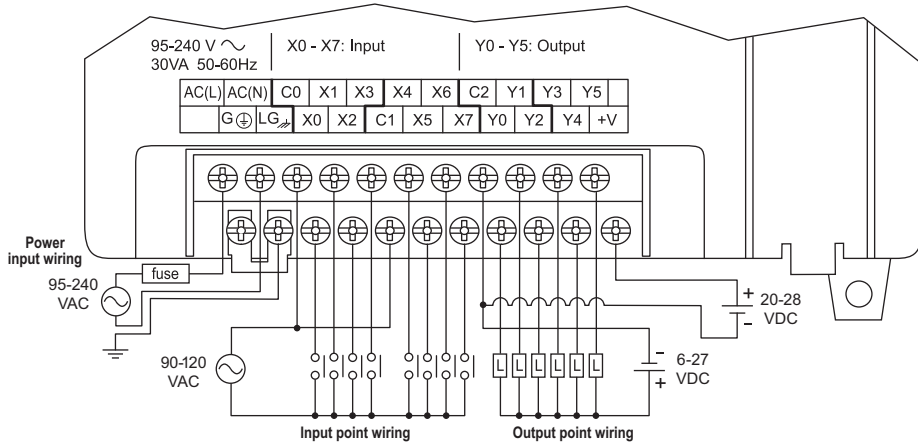
DO-05DR General Specifications	
External Power Requirements	95 – 240 VAC, 30 VA maximum,
Communication Port 1, 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity	K–Sequence (Slave), <b>DirectNET</b> (Slave), Modbus (Slave)
Communication Port 2, 9600 baud (default), 8 data bits, 1 stop bit, odd parity	K–Sequence (Slave), <b>DirectNET</b> (Master/Slave), Modbus (Master/Slave) Non-sequence / print
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 16AWG or two 18AWG, 24AWG minimum

DC Input Specifications		
Parameter	High–Speed Inputs, X0 – X2	Standard DC Inputs X3 – X7
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 (5 kHz maximum frequency)	30 VDC
Minimum Pulse Width	100 $\mu$ s	N/A
ON Voltage Level	> 10 VDC	> 10 VDC
OFF Voltage Level	< 2.0 VDC	< 2.0 VDC
Input Impedance	1.8 k $\Omega$ @ 12 – 24 VDC	2.8 k $\Omega$ @ 12 – 24 VDC
Max. Input Current	6mA @12VDC, 13mA @24VDC	4mA @12VDC, 8.5mA @24VDC
Minimum ON Current	>5 mA	>4 mA
Maximum OFF Current	< 0.5 mA	<0.5 mA
OFF to ON Response	<100 $\mu$ s	2 – 8 ms, 4 ms typical
ON to OFF Response	< 100 $\mu$ s	2 – 8 ms, 4 ms typical
Status Indicators	Logic side	Logic side
Commons	4 channels/common x 2 bank	

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

### D0-05AD I/O Wiring Diagram

The D0-05AD Micro PLC features eight AC inputs and six DC outputs. The following diagram shows a typical field wiring example. The AC external power connection uses four terminals at the left as shown.



The eight AC input channels use terminals in the middle of the connector. Inputs are organized into two banks of four. Each bank has an isolated common terminal. The wiring example above 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.

The six current sinking DC output channels use terminals on the right side of the connector. All outputs actually share the same electrical common. Note the requirement for external power on the end (right-most) terminal. The equivalent output circuit shows one channel of the bank of six.

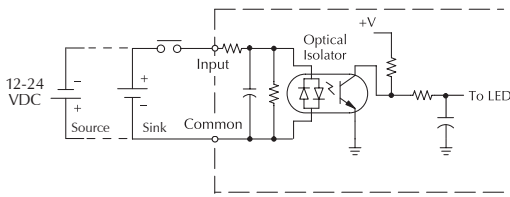
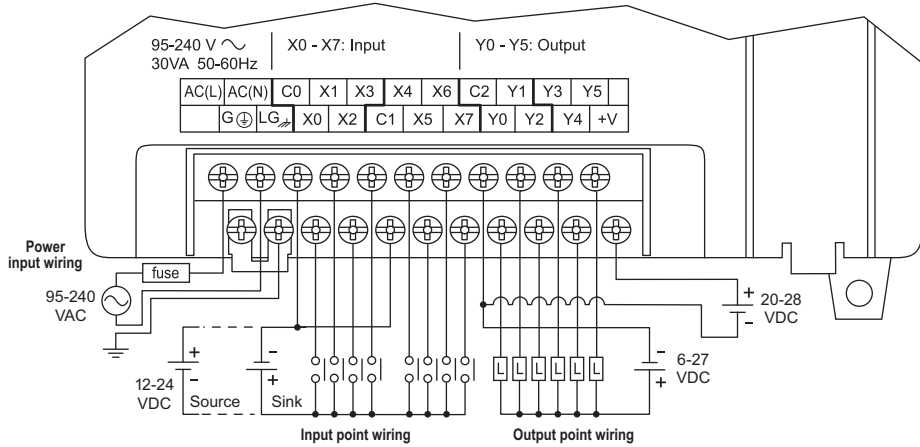
D0-05AD General Specifications	
External Power Requirements	95 – 240 VAC, 30 VA maximum,
Communication Port 1, 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), <b>DirectNET</b> (Slave), Modbus (Slave)
Communication Port 2, 9600 baud (default), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), <b>DirectNET</b> (Master/Slave), Modbus (Master/Slave) Non-sequence/print
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 16AWG or two 18AWG, 24AWG minimum

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

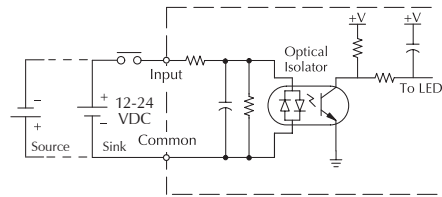
DC Output Specifications		
Parameter	Pulse Outputs, Y0 – Y1	Standard Outputs, Y2 – Y5
Min. - Max. Voltage Range	5 – 30 VDC	5 – 30 VDC
Operating Voltage	6 – 27 VDC	6 – 27 VDC
Peak Voltage	< 50 VDC (7 kHz max. frequency)	< 50 VDC
On Voltage Drop	0.3 VDC @ 1A	0.3 VDC @ 1A
Max Current (resistive)	0.5 A/pt. (1A/point for standard pt.)	1.0 A/point
Max leakage current	15 $\mu$ A @ 30 VDC	15 $\mu$ A @ 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 $\mu$ s	< 10 $\mu$ s
ON to OFF Response	<30 $\mu$ s	< 60 $\mu$ s
Status Indicators	Logic Side	Logic Side
Commons	6 channels/common x 1 bank	
Fuses	None	None

## D0-05DD I/O Wiring Diagram

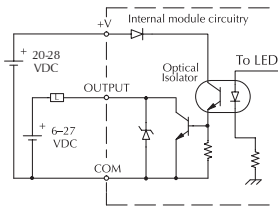
These micro PLCs feature eight DC inputs and six DC outputs. The following diagram shows a typical field wiring example. The AC external power connection uses four terminals at the left as shown.



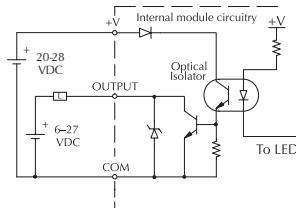
Equivalent Circuit, High Speed Input (X0-X2)



Equivalent Circuit, Standard Input (X3-X7)



Equivalent Circuit, Standard Output (Y2-Y5)

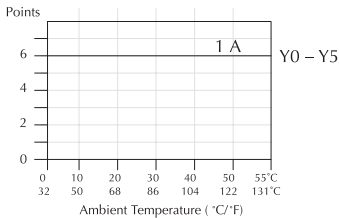


Equivalent Circuit, DC Pulse Output (Y0-Y1)

The eight DC input channels use terminals in the middle of the connector. Inputs are organized into two 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. The equivalent circuits for standard inputs and the high-speed inputs are shown above.

The six current sinking DC output channels use terminals on the right side of the connector. All outputs actually share the same electrical common. Note the requirement for external power on the end (right-most) terminal. The equivalent output circuit shows one channel of the bank of six.

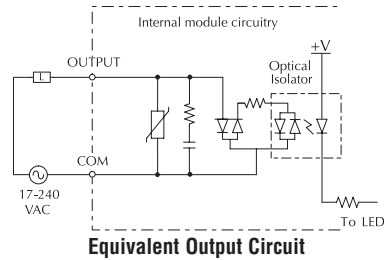
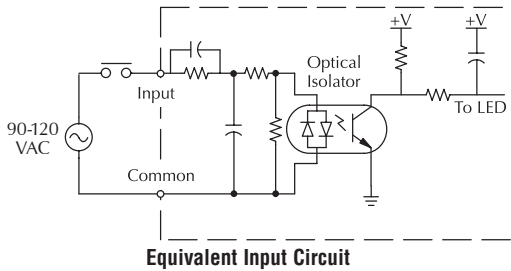
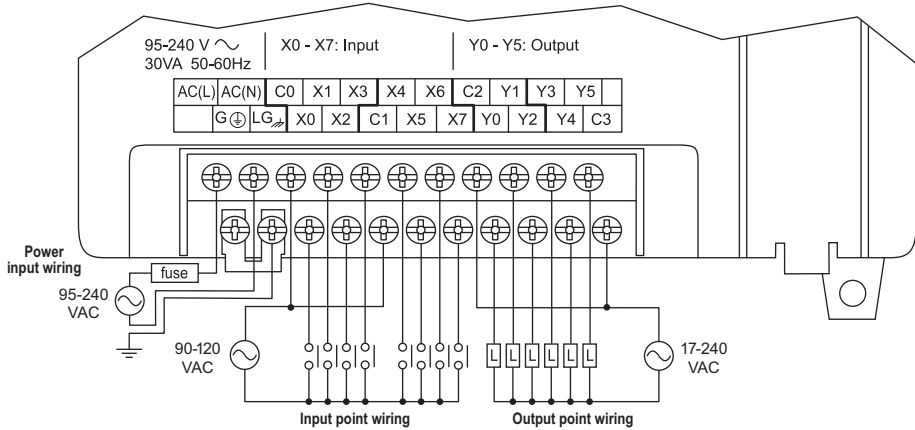


Derating Chart for DC Outputs

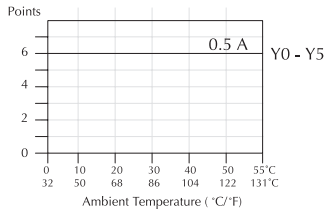
D0-05DD General Specifications		
External Power Requirements	95 – 240 VAC, 30 VA maximum,	
Communication Port 1, 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), <b>DirectNET</b> (Slave), Modbus (Slave)	
Communication Port 2, 9600 baud (default), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), <b>DirectNET</b> (Master/Slave), Modbus (Master/Slave) Non-sequence/print	
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	One16 AWG or two 18 AWG, 24 AWG minimum	
DC Input Specifications		
Parameter	High-Speed Inputs, X0 – X2	Standard DC Inputs X3 – X7
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 (5 kHz maximum frequency)	30 VDC
Minimum Pulse Width	100 µs	N/A
ON Voltage Level	> 9.0 VDC	> 9.0 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	<100 µs 2 – 8 ms	4 ms typical
ON to OFF Response	< 100 µs 2 – 8 ms	4 ms typical
Status Indicators	Logic side	Logic side
Commons	4 channels/common x 2 banks	
DC Output Specifications		
Parameter	Pulse Outputs Y0 – Y1	Standard Outputs Y3 – Y5
Min. - Max. Voltage Range	5 – 30 VDC	5 – 30 VDC
Operating Voltage	6 – 27 VDC	6 – 27 VDC
Peak Voltage	< 50 VDC (7 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 A @ 30 VDC	15 A @ 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	< 10µ s	< 10 µs
ON to OFF Response	< 30 µs	< 60 µs
Status Indicators	Logic Side	Logic Side
Commons	6 channels/common x 1 bank	
Fuses	None (external recommended)	

### D0-05AA I/O Wiring Diagram

The D0-05AA Micro PLC features eight AC inputs and six AC outputs. The following diagram shows a typical field wiring example. The AC external power connection uses four terminals at the left as shown.



The eight AC input channels use terminals in the middle of the connector. Inputs are organized into two banks of four. Each bank has an isolated common terminal. The wiring example above 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.



**Derating Chart for AC Outputs**

The six output channels use terminals on the right side of the connector. Outputs are organized into two banks of three triac switches. 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.

DO-05AA General Specifications	
External Power Requirements	95 – 240 VAC, 30 VA maximum,
Communication Port 1, 9600 baud (Fixed), 8 data bits, 1 stop bit odd parity	K-Sequence (Slave), <b>DirectNET</b> (Slave), Modbus (Slave)
Communication Port 2, 9600 baud (default) 8 data bits, 1 stop bit odd parity	K-Sequence (Slave), <b>DirectNET</b> (Master/Slave), Modbus (Master/Slave) Non-sequence/print
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 16AWG or two 18AWG, 24AWG minimum

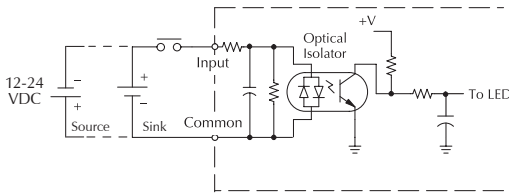
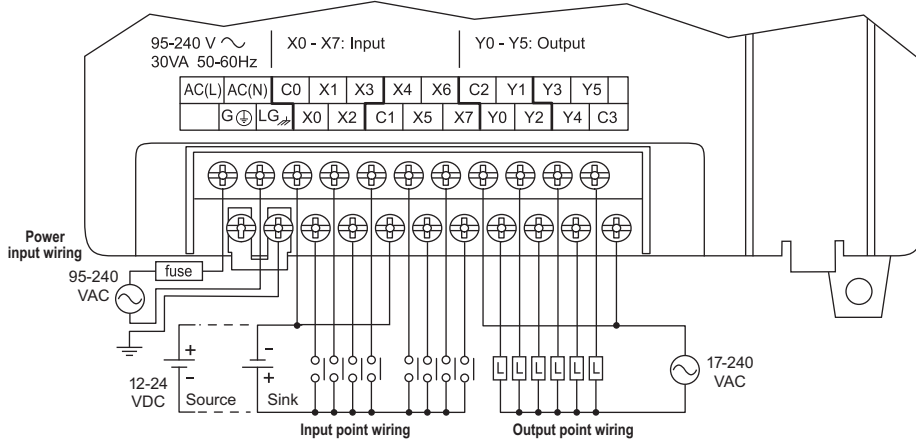
AC Input Specifications	
Input Voltage Range (Min. - Max.)	80 – 132 VAC, 47 - 63 Hz
Operating Voltage Range	90 – 120 VAC, 47 - 63 Hz
Input Current	8 mA @100 VAC at 50 Hz 10 mA @100 VAC at 60 Hz
Max. Input Current	12 mA @132 VAC at 50 Hz 15 mA @132 VAC at 60 Hz
Input Impedance	14 K $\Omega$ @ 50 Hz, 12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

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) 4.0 VAC (<50mA)
Max Current	0.5 A/point, 1.5 A/common
Max leakage current	<4 mA @ 264 VAC
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	3 channels/common x 2 banks
Fuses	None (external recommended)

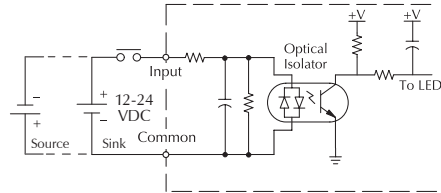
### D0-05DA I/O Wiring Diagram

The D0-05DA Micro PLC features eight DC inputs and six AC outputs. The following diagram shows a typical field wiring example. The AC external power connection uses four terminals at the left as shown.

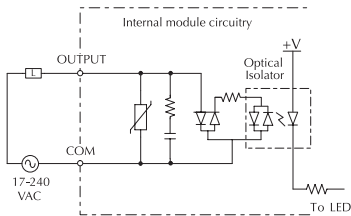
2



Equivalent Circuit, High Speed Input (X0-X2)

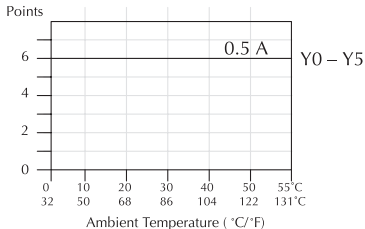


Equivalent Circuit, Standard Input (X3-X7)



Equivalent Circuit, Output

The eight DC input channels use terminals in the middle of the connector. Inputs are organized into two banks of four. Each bank has an isolated common terminal, and may be wired as sinking or sourcing inputs. The wiring example above shows all commons connected together, but separate supplies and common circuits may be used. The equivalent circuit for standard inputs and the high-speed input circuit are shown above.



Derating Chart for AC Outputs

The six output channels use terminals on the right side of the connector. Outputs are organized into two banks of three triac switches. 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.



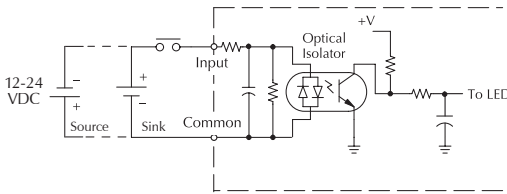
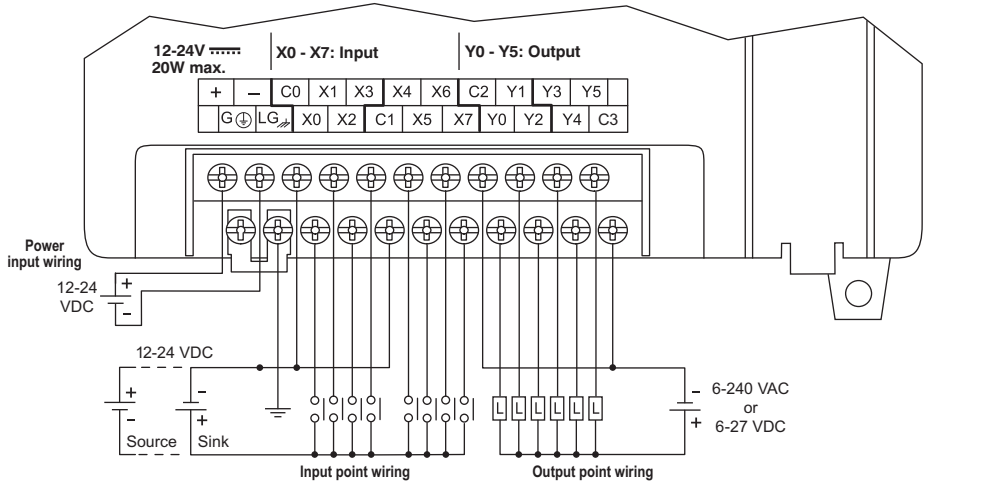
D0-05DA General Specifications	
External Power Requirements	95 – 240 VAC, 30 VA maximum,
Communication Port 1, 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), <b>DirectNET</b> (Slave), Modbus (Slave)
Communication Port 2, 9600 baud (default), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), <b>DirectNET</b> (Master/Slave), Modbus (Master/Slave) Non-sequence/print
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 16 AWG or two 18 AWG, 24AWG minimum

DC Input Specifications		
Parameter	High-Speed Inputs, X0 – X2	Standard DC Inputs X3 – X7
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 (5 kHz maximum frequency)	30 VDC
Minimum Pulse Width	100 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	<100 µs	2 – 8 ms, 4 ms typical
ON to OFF Response	< 100 µs	2 – 8 ms, 4 ms typical
Status Indicators	Logic side	Logic side
Commons	4 channels / common x 2 banks	

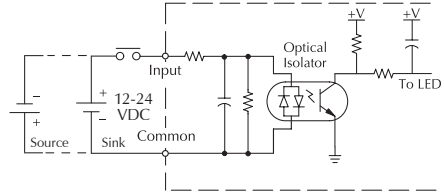
AC Output Specifications	
Output Voltage Range (Min. - Max.)	15 – 264 VAC, 47 – 63 Hz
Operating Voltage	17 – 240 VAC, 47 – 63 Hz
On Voltage Drop	1.5 VAC @> 50mA, 4 VAC @< 50mA
Max Current	0.5 A/point, 1.5 A/common
Max leakage current	< 4 mA @ 264 VAC, 60Hz
Max inrush current	10 A for 10 ms
Minimum Load	10 mA
OFF to ON Response	1 ms
ON to OFF Response	1 ms +1/2 cycle
Status Indicators	Logic Side
Commons	3 channels / common x 2 banks
Fuses	None (external recommended)

## D0-05DR-D I/O Wiring Diagram

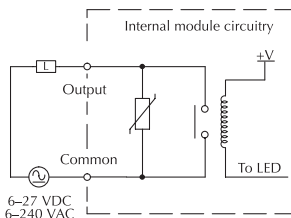
These micro PLCs feature eight DC inputs and six relay contact outputs. The following diagram shows a typical field wiring example. The DC external power connection uses three terminals at the left as shown.



Equivalent Circuit, High-speed Input (X0-X2)

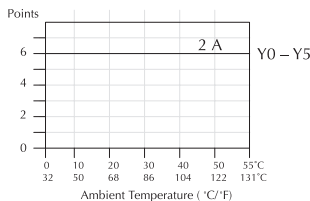


Equivalent Circuit, Standard Input (X3-X7)



Equivalent Circuit, Standard Output

The eight DC input channels use terminals in the middle of the connector. Inputs are organized into two 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. The equivalent circuit for standard inputs and the high-speed input circuit are shown above.



Derating Chart for Relay Outputs

The six output channels use terminals on the right side of the connector. Outputs are organized into two banks of three 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.

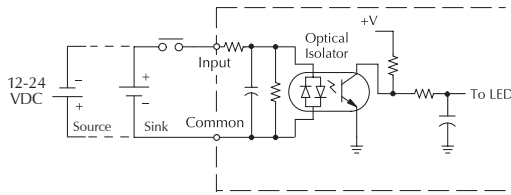
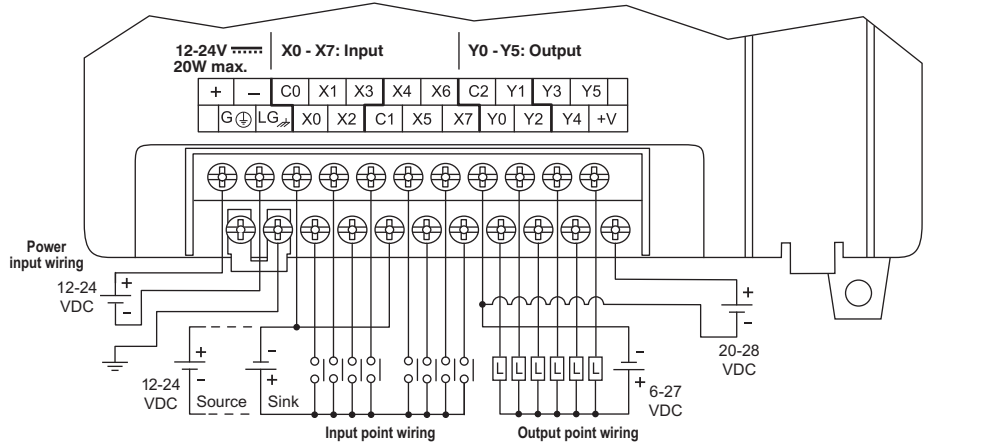
DO-05DR-D General Specifications	
External Power Requirements	12 – 24 VDC, 20 W maximum,
Communication Port 1, 9600 baud (Fixed), 8 data bits, 1 stop bit, odd parity	K–Sequence (Slave), <b>DirectNET</b> (Slave), Modbus (Slave)
Communication Port 2, 9600 baud (default), 8 data bits, 1 stop bit, odd parity	K–Sequence (Slave), <b>DirectNET</b> (Master/Slave), Modbus (Master/Slave) Non-sequence/print
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 16AWG or two 18AWG, 24AWG minimum

DC Input Specifications		
Parameter	High–Speed Inputs, X0 – X2	Standard DC Inputs X3 – X7
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 (5 kHz maximum frequency)	30 VDC
Minimum Pulse Width	100 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
Max. Input Current	6mA @12VDC ,13mA @24VDC	4mA @12VDC, 8.5mA @24VDC
Minimum ON Current	>5 mA	>4 mA
Maximum OFF Current	< 0.5 mA	<0.5 mA
OFF to ON Response	<100 µs	2 – 8 ms, 4 ms typical
ON to OFF Response	< 100 µs	2 – 8 ms, 4 ms typical
Status Indicators	Logic side	Logic side
Commons	4 channels / common x 2 banks	

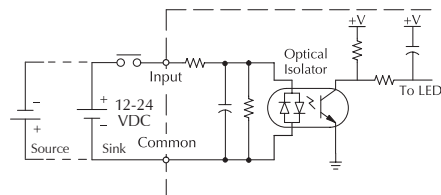
Relay Output Specifications	
Output Voltage Range (Min. - Max.)	5 -264 VAC (47 -63 Hz), 5 - 30 VDC
Operating Voltage	6 -240 VAC (47 -63 Hz), 6 - 27 VDC
Output Current	2A/point 6A/common
Maximum Voltage	264 VAC, 30 VDC
Max leakage current	0.1 mA @264 VAC
Smallest Recommended Load	5 mA
OFF to ON Response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

### D0-05DD-D I/O Wiring Diagram

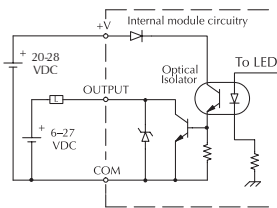
These micro PLCs feature eight DC inputs and six DC outputs. The following diagram shows a typical field wiring example. The DC external power connection uses four terminals at the left as shown.



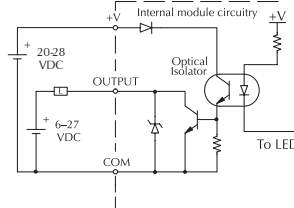
Equivalent Circuit, High Speed Input (X0-X2)



Equivalent Circuit, Standard Input (X3-X7)



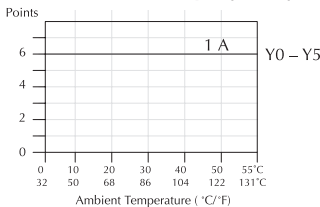
Equivalent Circuit, Standard Output (Y2-Y5)



Equivalent Circuit, Pulse Output (Y0-Y1)

The eight DC input channels use terminals in the middle of the connector. Inputs are organized into two 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. The equivalent circuit for standard inputs and the high-speed input circuit are shown above.

The six current sinking DC output channels use terminals on the right side of the connector. All outputs actually share the same electrical common. Note the requirement for external power on the end (right-most) terminal. The equivalent output circuit shows one channel of the bank of six.



Derating Chart for DC Outputs

D0-05DD-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), <b>DirectNET</b> (Slave), Modbus RTU (Slave)
Communication Port 2, 9600 baud (default), 8 data bits, 1 stop bit, odd parity	K-Sequence (Slave), <b>DirectNET</b> (Master/Slave), Modbus RTU (Master/Slave) Non-sequence/print
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 16AWG or two 18AWG, 24AWG minimum

DC Input Specifications		
Parameter	High-Speed Inputs, X0 – X2	Standard DC Inputs X3 – X7
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 (5 kHz maximum frequency)	30 VDC
Minimum Pulse Width	100 µs	N/A
ON Voltage Level	>9.0 VDC	> 9.0 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	<100 µs	2 – 8 ms, 4 ms typical
ON to OFF Response	< 100 µs	2 – 8 ms, 4 ms typical
Status Indicators	Logic side	Logic side
Commons	4 channels / common x 2 banks	

DC Output Specifications		
Parameter	Pulse Outputs, Y0 – Y1	Standard Outputs, Y3 – Y5
Min. - Max. Voltage Range	5 – 30 VDC	5 – 30 VDC
Operating Voltage	6 – 27 VDC	6 – 27 VDC
Peak Voltage	< 50 VDC (7 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	< 30 µs	< 60 µs
Status Indicators	Logic Side	Logic Side
Commons	6 channels / common x 1 bank	
Fuses	None (external recommended)	

## Glossary of Specification Terms

### **Discrete Input**

One of eight 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 six 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 DL05 Micro PLCs are electrically located on the logic side of the input or output circuit.

# CPU SPECIFICATIONS AND OPERATION

---



# CHAPTER 3

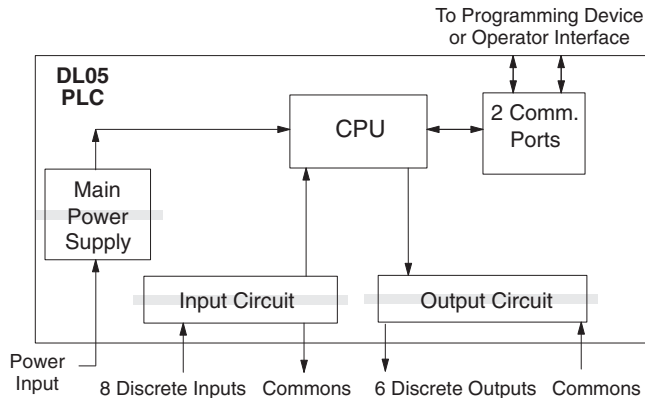
## In This Chapter:

Introduction . . . . .	3-2
CPU Specifications . . . . .	3-3
CPU Hardware Setup . . . . .	3-4
CPU Operation . . . . .	3-11
I/O Response Time . . . . .	3-15
CPU Scan Time Considerations . . . . .	3-18
Memory Map . . . . .	3-22
DL05 System V-memory . . . . .	3-26
DL05 Aliases . . . . .	3-29
X Input Bit Map . . . . .	3-30
Y Output Bit Map . . . . .	3-30
Control Relay Bit Map . . . . .	3-31
Stage Control/Status Bit Map . . . . .	3-32
Timer Status Bit Map . . . . .	3-32
Counter Status Bit Map . . . . .	3-33

## 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 DL05. It is not considered part of the CPU, because it does not execute the ladder program. For more on HSIO operation, see Appendix E.

### DL05 CPU Features

The DL05 Micro PLC which has 6K words of memory comprised of 2K of ladder memory and 4K 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 DL05 supports fixed I/O which includes eight discrete input points and six output points. If more than the fourteen fixed I/O points are needed, select an I/O module for your application from the DL05/06 Option Modules User Manual. This module will plug into the expansion slot.

Over 120 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 DL05 provides two built-in RS232C 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	DL05
Total Program memory (words)	6K
Ladder memory (words)	2048
Total V-memory (words)	4096
User V-memory (words)	3968
Non-volatile V-Memory (words)	128
Contact execution (boolean)	2.0us
Typical scan (1k boolean)	2.7–3.2ms
RLL Ladder Style Programming	Yes
RLL and RLL <sup>PLUS</sup> Programming	Yes
Run Time Edits	Yes
Supports Overrides	Yes
Scan	Variable / fixed
Handheld programmer	Yes
<i>DirectSOFT 5</i> programming for Windows.	Yes
Built-in communication ports (RS232C)	Yes
FLASH Memory	Standard on CPU
Local Discrete I/O points available	14
Local Analog input / output channels maximum	None
High-Speed I/O (quad., pulse out, interrupt, pulse catch, etc.)	Yes, 2
I/O Point Density	8 inputs, 6 outputs
Number of instructions available (see Chapter 5 for details)	129
Control relays	512
Special relays (system defined)	512
Stages in RLL <sup>PLUS</sup>	256
Timers	128
Counters	128
Immediate I/O	Yes
Interrupt input (external/timed)	Yes
Subroutines	Yes
For/Next Loops	Yes
Math	Integer
Drum Sequencer Instruction	Yes
Time of Day Clock/Calendar	Only with the optional Memory Cartridge
Internal diagnostics	Yes
Password security	Yes
System error log	No
User error log	No
Battery backup	No (built-in super-cap) Yes, with memory cartridge

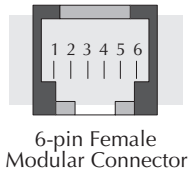
# CPU Hardware Setup

## Communication Port Pinout Diagrams

Cables are available that allow you to quickly and easily connect a Handheld Programmer or a personal computer to the DL05 PLCs. However, if you need to build your own cables, use the pinout information shown below. The DL05 PLCs require an RJ-12 phone plug to fit the built-in jacks.

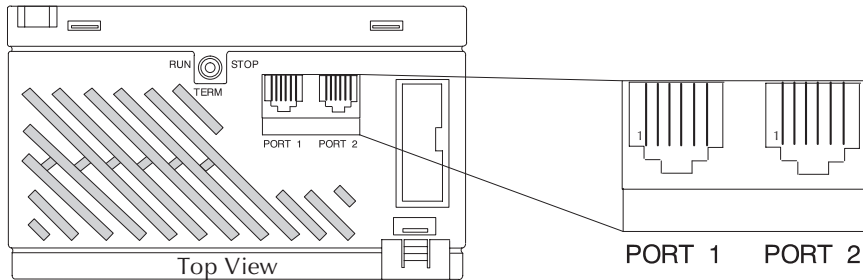
The Micro PLC has two built-in RS232C communication ports. Port 1 is generally used for connecting to a D2-HPP, a PC with *DirectSOFT*, operator interface, Modbus slave, or a *DirectNET* slave. The baud rate is fixed at 9600 baud. Port 2 can be used to connect to a D2-HPP, *DirectSOFT*, operator interface, Modbus master/slave, or a *DirectNET* master/slave. Port 2 has a range of speeds from 300 baud to 38.4K baud.

**NOTE:** The 5V pins are rated at 220mA maximum, primarily for use with some operator interface units.



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

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

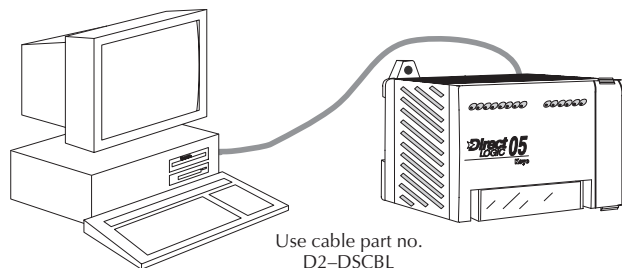


Communication Port 1	
Com 1	Connects to HPP, <i>DirectSOFT</i> , operator interfaces, etc. 6-pin, RS232C 9600 Baud (Fixed) Parity - odd (default) Station address 1 (fixed) 8 data bits 1 start, 1 stop bit Asynchronous, Half-duplex, DTE Protocol: (Auto-Select) K sequence (Slave only) <i>DirectNET</i> (Slave only) Modbus RTU (Slave only)

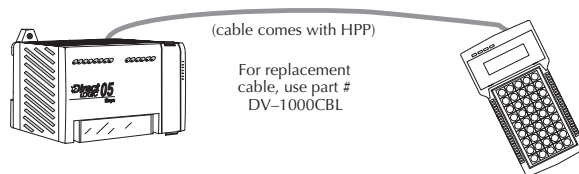
Communication Port 2	
Com 2	Connects to HPP, <i>DirectSOFT</i> , operator interfaces, etc. 6-pin, RS232C Communication speed (baud) 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 Parity - odd (default), even, none Station address 1 (default) 8 data bits 1 start, 1 stop bit Asynchronous, Half-duplex, DTE Protocol: (Auto-Select) K sequence (Slave only) <i>DirectNET</i> (Master/Slave) Modbus RTU (Master/Slave) Non-sequence/Print

## Connecting the Programming Devices

If you're using a Personal Computer with the *DirectSOFT 5* programming package, you can connect the computer to either of the DL05's programming ports. For an engineering office environment (typical during program development), this is the preferred method of programming.



The Handheld programmer is connected to the CPU with a handheld programmer cable. This device can be used for maintaining existing installations or making small program changes whenever a PC is not available. The handheld programmer is shipped with a cable, which is approximately 6.5 feet (200 cm) long.

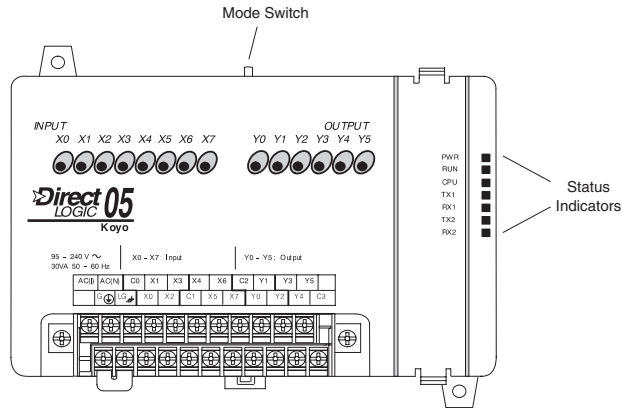


## CPU Setup Information

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. Selecting and Changing the CPU Modes

- 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 *DirectSOFT 5* Programming Software User Manual provides a description of the menus and keystrokes required to perform the setup procedures.



## Status Indicators

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

Indicator	Status	Meaning
PWR	ON	Power good
	OFF	Power failure
RUN	ON	CPU is in Run Mode
	OFF	CPU is in Stop or program Mode
	Blinking	CPU is in upgrade Mode
CPU	ON	CPU self diagnostics error
	OFF	CPU self diagnostics good
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

## Mode Switch Functions

The mode switch on the DL05 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, *DirectSOFT 5* 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.

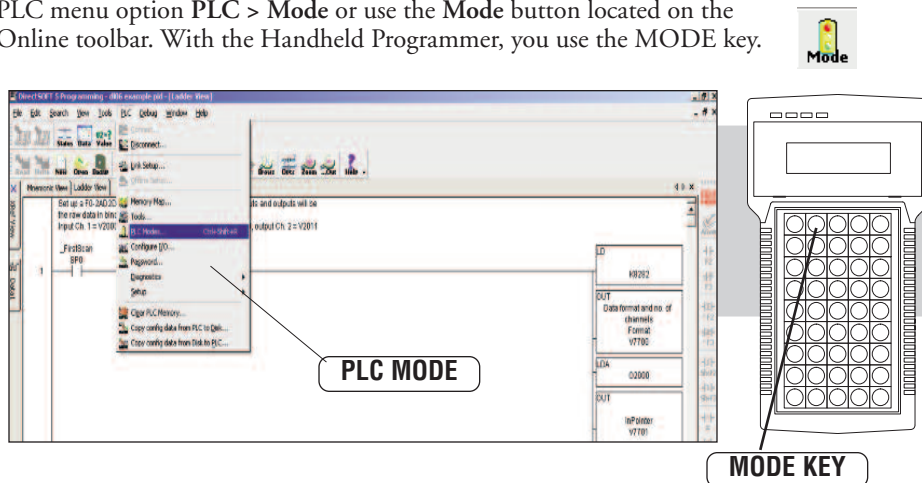


**NOTE:** If the DL05 is switched to the RUN Mode without a program in the PLC, the PLC will produce a FATAL ERROR which can be cleared by cycling power to the PLC.

## Changing Modes in the DL05 PLC

Modeswitch 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 *DirectSOFT 5* or the Handheld Programmer to change the CPU mode of operation. With *DirectSOFT 5* use the PLC menu option **PLC > Mode** or use the **Mode** button located on the Online toolbar. With the Handheld Programmer, you use the **MODE** key.



### Mode of Operation at Power-up

The DL05 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 may not power-up in the mode it was in when this occurred. There is no way to determine which mode will be entered as the startup mode. However, the PLC can power-up in either Run or Program Mode if the mode switch is in the TERM position. Failure to adhere to this warning greatly increases the risk of unexpected equipment startup.

The mode which the CPU will power-up in is also determined by the state of B7633.13. If the bit is set and the Mode Switch is in the TERM position, then the CPU will power-up in the state it was in at power-down.

### 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 *DirectSOFT 5*, 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 *DirectSOFT 5* package. The following table shows a list of the Auxiliary functions for the Handheld Programmer.

<b>AUX 2* — RLL Operations</b>		5B	HSIO Configuration
21	Check Program	5D	Scan Control Setup
22	Change Reference	<b>AUX 6* — Handheld Programmer Configuration</b>	
23	Clear Ladder Range	61	Show Revision Numbers
24	Clear All Ladders	62	Beeper On / Off
<b>AUX 3* — V-Memory Operations</b>		65	Run Self Diagnostics
31	Clear V-Memory	<b>AUX 7* — EEPROM Operations</b>	
<b>AUX 4* — I/O Configuration</b>		71	Copy CPU memory to HPP EEPROM
41	Show I/O Configuration	72	Write HPP EEPROM to CPU
<b>AUX 5* — CPU Configuration</b>		73	Compare CPU to HPP EEPROM
51	Modify Program Name	74	Blank Check (HPP EEPROM)
53	Display Scan Time	75	Erase HPP EEPROM
54	Initialize Scratchpad	76	Show EEPROM Type (CPU and HPP)
55	Set Watchdog Timer	<b>AUX 8* — Password Operations</b>	
56	Set Communication Port 2	81	Modify Password
57	Set Retentive Ranges	82	Unlock CPU
58	Test Operations	83	Lock CPU
59	Override Setup		

### 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 DL05 Micro PLC maintain system parameters in a memory area often referred to as the “scratchpad”. In some cases, you may make changes to the system setup that will be stored in system memory. For example, if you specify a range of Control Relays (CRs) as retentive, these changes are stored 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 you that you have considered all ramifications of this operation before you select it.

### Setting Retentive Memory Ranges

The DL05 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. Appendix F has more information pertaining to the different types of memory. The default settings are:

Memory Area	DL05	
	Default Range	Available Range
Control Relays	C400 – C777	C0 – C777
V-memory	V1400 – V7777	V0 – V7777
Timers	None by default	T0 – T177
Counters	CT0 – CT177	CT0 – CT177
Stages	None by default	S0 – S377

You can use AUX 57 to set the retentive ranges. Appendix A contains detailed information about auxiliary functions. You can also set the retentive ranges by using Setup in *DirectSOFT* 5, PLC > Setup > Retentive Ranges.



**WARNING:** The DL05 PLCs do not have battery back-up (unless the memory cartridge, D0-01MC, is installed) 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.

### Using a Password

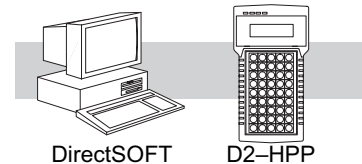
The DL05 PLCs allow you to use a password to help minimize the risk of unauthorized program and/or data changes. Once you enter a password you can “lock” the PLC against access. Once the CPU is locked you must enter the password before you can use a programming device to change any system parameters.

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) cleared from memory. It is the policy of AutomationDirect to require the memory of the PLC to be cleared along with the password.**

You can use the D2–HPP Handheld Programmer or *DirectSOFT 5*. to enter a password. The following diagram shows how you can enter a password with the Handheld Programmer.



Select AUX 81



Enter the new 8-digit password



Press CLR to clear the display

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 *DirectSOFT 5*, the CPU will be automatically locked against access when you exit *DirectSOFT 5*.
3. Use AUX 83 to lock the CPU.

When you use *DirectSOFT 5*, 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.



## CPU Operation

Achieving the proper control for your equipment or process requires a good understanding of how DL05 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 — DL05 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.

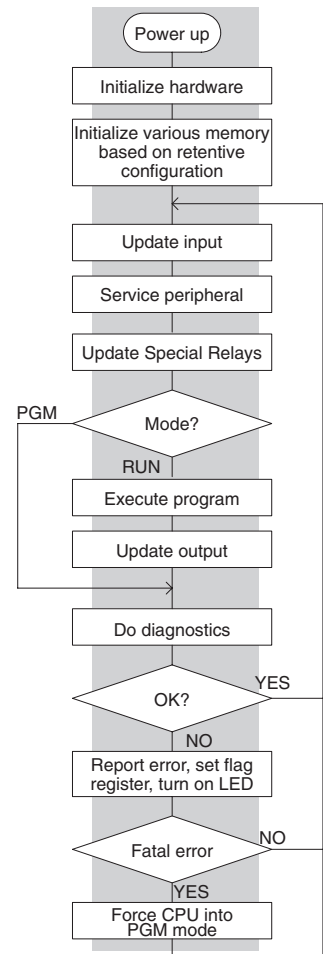
### CPU Operating System

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

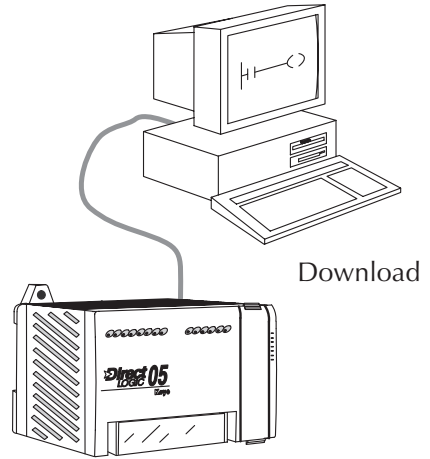
Error detection has two levels. Non-fatal errors are reported, but the CPU remains in its current mode. If a fatal error occurs, the CPU is forced into program mode and the outputs go off.



## 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 a PC with *DirectSOFT 5* Programming Software or the D2–HPP Handheld programmer to place the CPU in Program Mode.



## Run Mode

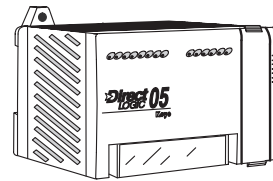
In Run Mode, the CPU executes the application program and updates the I/O system. You can perform many operations during Run Mode. Some of these include:

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

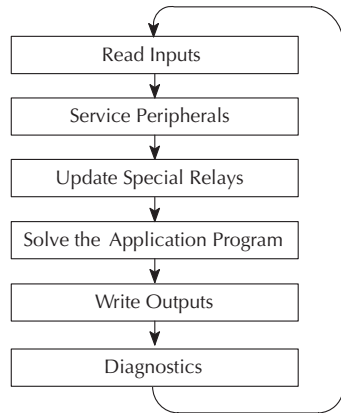
Run Mode operation can be divided into several key areas. For the vast majority of applications, some of these execution segments are more important than others. For example, you need to understand how the CPU updates the I/O points, handles forcing operations, and solves the application program. The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

You can use *DirectSOFT 5* 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.



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

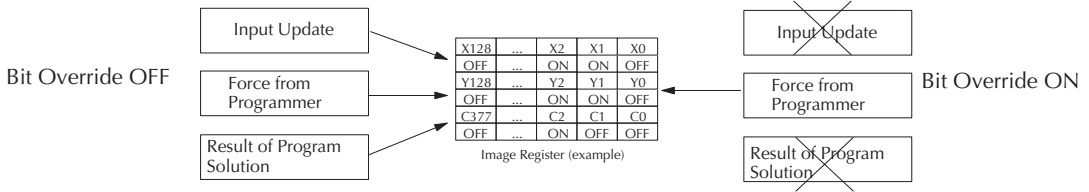
After the CPU reads the inputs from the input modules, it reads any attached peripheral devices. This is primarily a communications service for any attached devices. For example, it would read a programming device to see if any input, output, or other memory type status needs to be modified. There are two basic types of forcing available with the DL05 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 using the Data View option within *DirectSOFT 5*. 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.**

## Update Special Relays and Special Registers

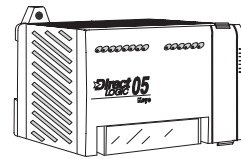
There are dedicated V-memory locations that contain Special Relays and other dedicated register information. This portion of the execution cycle makes sure these locations get updated on every scan. Also, there are several different Special Relays, such as diagnostic relays, etc., that are also updated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

## Solve Application Program

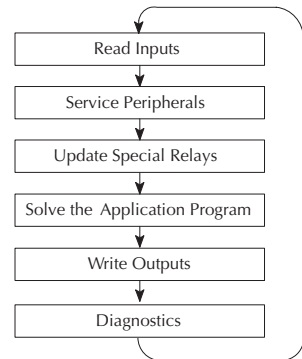
The CPU evaluates each instruction in the application program during this segment of the scan cycle. The instructions define the relationship between the input conditions and the desired output response. The CPU uses the output image register area to store the status of the desired action for the outputs. 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.



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.

## 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 DL05 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 DL05 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.

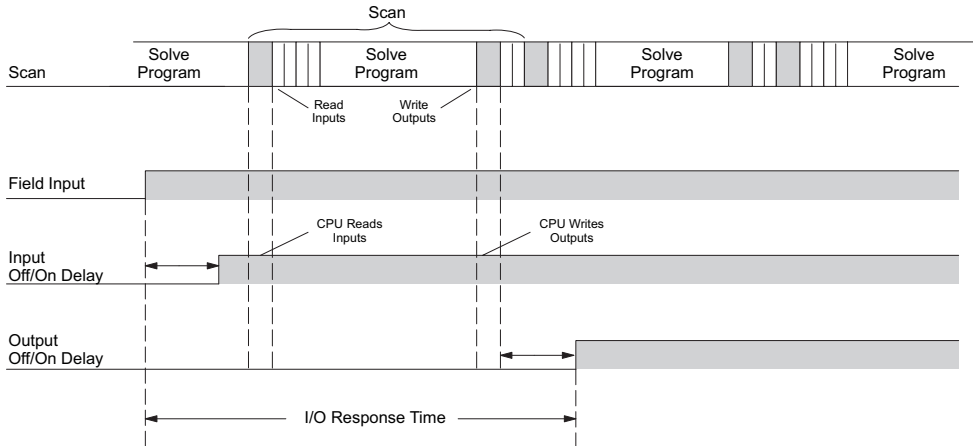
There are four things that can affect the I/O response time.

- 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.

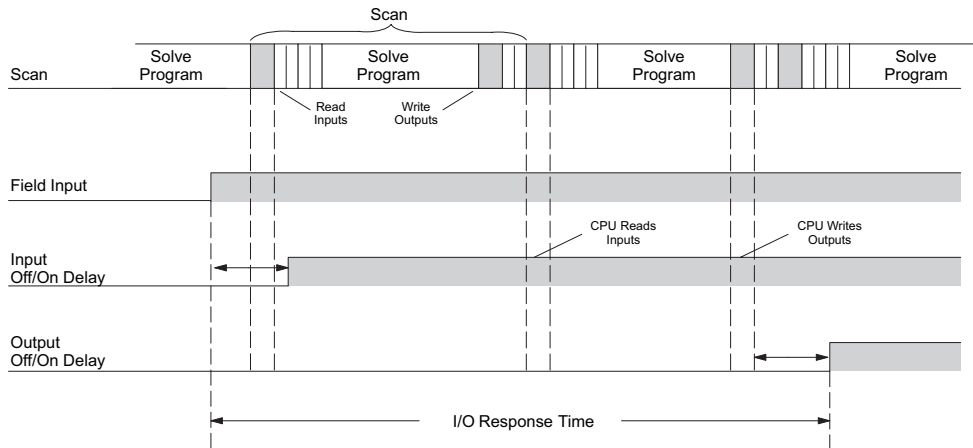


In this case, you can calculate the response time by simply adding the following items:

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

### 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:

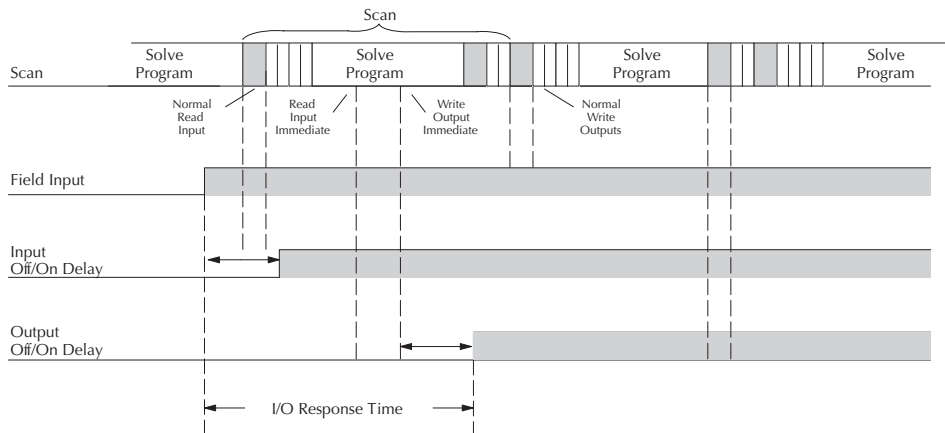
$$\text{Input Delay} + (2 \times \text{Scan Time}) + \text{Output Delay} = \text{Response Time}$$

## 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 Appendix E for details on using this feature.
- Change Mode 60 filter to 0 msec for X0, X1, X2 and X3.

Of these four 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

3

The scan time covers all the cyclical tasks that are performed by the operating system. You can use *DirectSOFT 5* 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 only one you really have the most control over is the amount of time it takes to execute the application program. This is because different instructions take different amounts of time to execute. So, if you think you need a faster scan, then you can try to choose faster instructions.

Your choice of I/O type and peripheral devices can also affect the scan time. However, these things are usually dictated by the application.

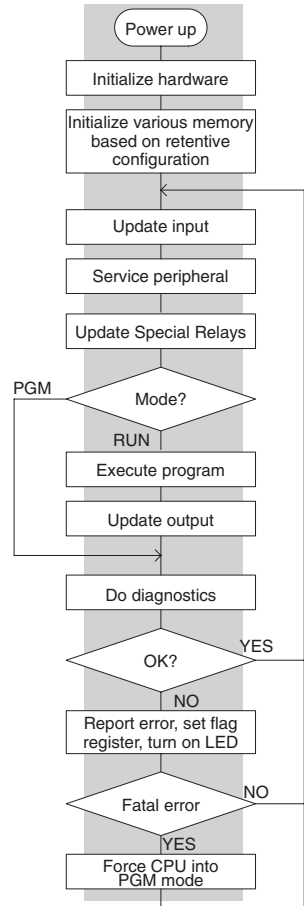
The following paragraphs provide some general information on how much time some of the segments can require.

### Reading Inputs

The time required during each scan to read the input status is 40  $\mu$ 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 is 629  $\mu$ s. Don't confuse this with the I/O response time that was discussed earlier.



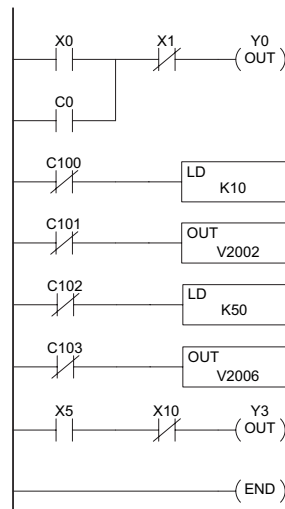


## Application Program Execution

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 DL05 Micro PLC. For example, the execution time for running the program shown below is calculated as follows:

Instruction	Time
STR X0	2.0 $\mu$ s
OR C0	1.6 $\mu$ s
ANDN X1	1.6 $\mu$ s
OUT Y0	6.8 $\mu$ s
STRN C100	2.3 $\mu$ s
LD K10	42.7 $\mu$ s
STRN C101	2.3 $\mu$ s
OUT V2002	16.6 $\mu$ s
STRN C102	2.3 $\mu$ s
LD K50	42.7 $\mu$ s
STRN C103	2.3 $\mu$ s
OUT V2006	16.6 $\mu$ s
STR X5	2.0 $\mu$ s
ANDN X10	1.6 $\mu$ s
OUT Y3	6.8 $\mu$ s
END	24.0 $\mu$ s
<b>SUBTOTAL</b>	<b>174.2 <math>\mu</math>s</b>
<b>Overhead DL05</b>	
Minimum	0.66 $\mu$ s
Maximum	2.5 $\mu$ s



$$\text{TOTAL TIME} = (\text{Program execution time} + \text{Overhead}) \times 1.1$$

The program above takes only 174.2  $\mu$ s to execute during each scan. The DL05 spends 0.1 ms, on internal timed interrupt management, for every 1.0 ms 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 1.1. “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 DL05 PLCs have an interrupt routine feature that changes the way a program executes. Since this instruction interrupts normal program flow, it will have an effect on the program execution time. For example, a timed interrupt routine with a 10.0 ms period interrupts the main program execution (before the END statement) every 10.0 ms, so the CPU can execute the interrupt routine. Chapter 5 provides detailed information on interrupts.

### PLC Numbering Systems

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

octal	1482	BCD	?	binary	
?		?	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 (see Appendix I for numbering system details).

### PLC Resources

PLCs offer a fixed amount of resources, depending on the model and configuration. We use the word “resources” to include variable memory (V-memory), I/O points, timers, counters, etc. Most modular PLCs allow you to add I/O points in groups of eight. In fact, all the resources of our PLCs are counted in octal. It's easier for computers to count in groups of eight than ten, because eight is an even power of 2.

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

Decimal	1	2	3	4	5	6	7	8
	●	●	●	●	●	●	●	●
Octal	1	2	3	4	5	6	7	10

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.

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

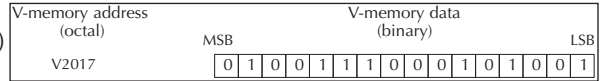
After counting PLC resources, it is time to access PLC resources (there is a difference). The CPU instruction set accesses resources of the PLC using octal addresses. Octal addresses are the same as octal quantities, except they start counting at zero. The number zero is significant to a computer, so 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.

X=	0	1	2	3	4	5	6	7
X	●	●	●	●	●	●	●	●
1 X	●	●	●	●	●	●	●	●
2 X	●	●	●	●	●	●	●	●

### 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).



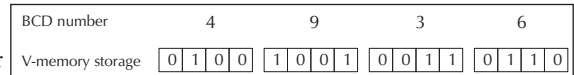
Each V-memory location is one data word wide, meaning 16 bits. For configuration registers, our manuals will show each bit of a V-memory word. The least significant bit (LSB) will be on the right, and the most significant bit (MSB) on the left. We use the word “significant”, referring to the relative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

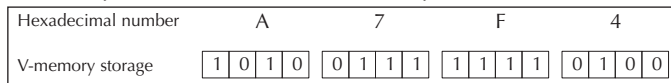
In a pure binary sense, a 16-bit word can represent numbers from 0 to 65535. In storing BCD numbers, the range is reduced to only 0 to 9999. Many math instructions use Binary-Coded Decimal (BCD) data, and *DirectSOFT* 5 and the handheld programmer allow us to enter and view data in BCD.

### 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.



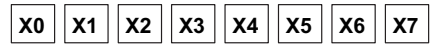
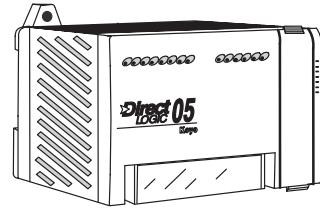
## Memory Map

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 DL05 Micro PLCs. A memory map overview for the CPU follows the memory descriptions.

3

### 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

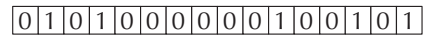


### Discrete and Word Locations

As you examine the different memory types, you'll notice two types of memory in the DL05, 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.

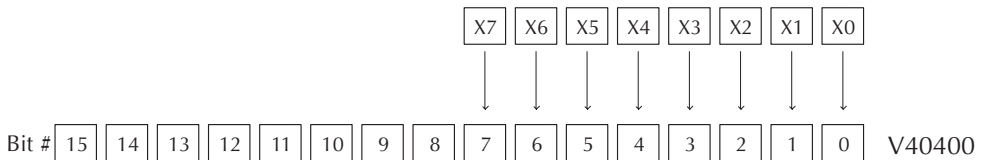
Word Locations – 16 bits



### 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 DL05 Micro PLCs on the following pages.

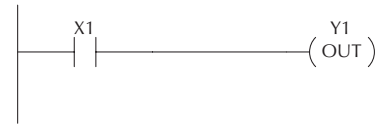
### Input Points (X Data Type)

The discrete input points are noted by an X data type. There are 8 discrete input points and 256 discrete input addresses available with DL05 CPUs. In this example, the output point Y0 will be turned on when input X0 energizes.



### Output Points (Y Data Type)

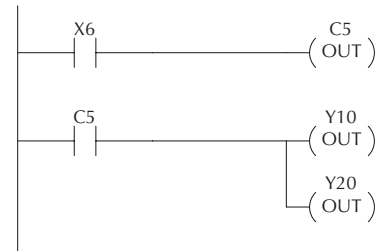
The discrete output points are noted by a Y data type. There are 6 discrete outputs and 256 discrete output addresses available with DL05 CPUs. In this example, output point Y1 will be turned on when input X1 energizes.



### 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.

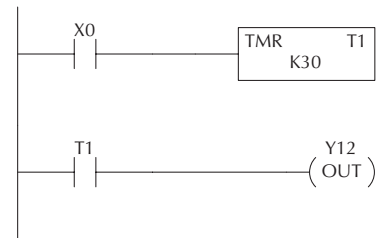
In this example, memory location C5 will energize when input X6 turns on. The second rung shows a simple example of how to use a control relay as an input.



### 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.

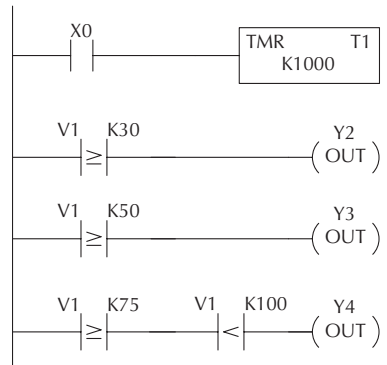
When input X0 turns on, timer T1 will start. When the timer reaches the preset of 3 seconds (K30) timer status contact T1 turns on. When T1 turns on, output Y12 turns on. Turning off X0 resets the timer.



### 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.

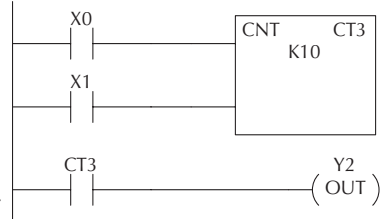
The primary reason for this is programming flexibility. The example shows how you can use relational contacts of a monitor several time intervals from a single timer.



### 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.

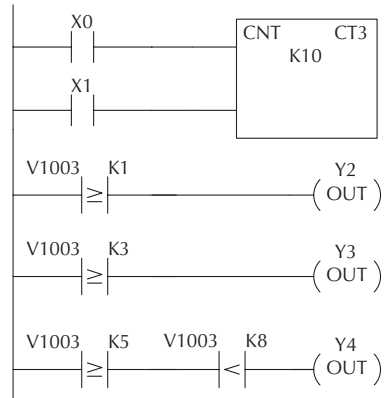
Each time contact X0 transitions from off to on, the counter increments by one. (If X1 comes on, the counter is reset to zero.) When the counter reaches the preset of 10 counts (K of 10) counter status contact CT3 turns on. When CT3 turns on, output Y2 turns on.



### Counter Current Values (V Data Type)

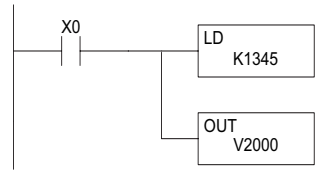
Just like the timers, the counter current values are also automatically stored in V-memory. For example, V1000 holds the current value for Counter CT0, V1001 holds the current value for Counter CT1, etc.

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)

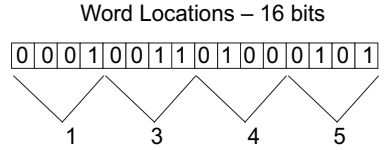
Word memory is referred to as V-memory (variable) and is a 16-bit location normally used to manipulate data/numbers, store data/numbers, etc. Some information is automatically stored in V-memory. For example, the timer current values are stored in V-memory. The example shows how a four-digit BCD constant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



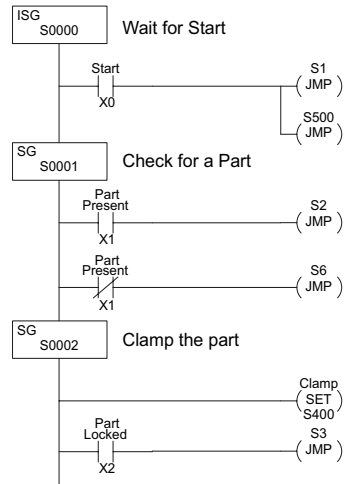
### Stages (S Data type)

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.



### 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

## DL05 System V-memory

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

The DL05 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.

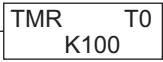
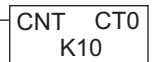

3

System V-memory	Description of Contents	Default Values/Ranges
<b>V2320–V2377</b>	The default location for multiple preset values for the High-Speed Counter	N/A
<b>V7620–V7627</b>	<b>Locations for DV–1000 operator interface parameters</b>	
<b>V7620</b>	Sets the V-memory location that contains the value	V0 – V2377
<b>V7621</b>	Sets the V-memory location that contains the message	V0 – V2377
<b>V7622</b>	Sets the total number (1 – 16) of V-memory locations to be displayed.	1 – 16
<b>V7623</b>	Sets the V-memory location containing the numbers to be displayed.	V0 – V2377
<b>V7624</b>	Sets the V-memory location containing the character code to be displayed	V0 – V2377
<b>V7625</b>	Contains the function number that can be assigned to each key.	V-memory location for X, Y, or C points used
<b>V7626</b>	Power-up operational mode.	0, 1, 2, 12, 3
<b>V7627</b>	Change preset value.	0000 to 9999
<b>V7630</b>	Starting location for the multi-step presets for channel 1. The default value is 2320, which indicates the first value should be obtained from V2320. Since there are 24 presets available, the default range is V2320 – V2377. You can change the starting point if necessary.	Default: V2320 Range: V0 – V2320
<b>V7631–V7632</b>	Reserved	N/A
<b>V7633</b>	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 8–12, 14, 15: Unused Bit 13: Power-up in RUN, if Mode Switch is in TERM position.
<b>V7634 - X0</b>		
<b>V7635 - X1</b>	Setup Registers for High-Speed I/O functions	Default: 1006
<b>V7636 - X2</b>		
<b>V7637</b>	Pulse/Direction	Default: 0000
<b>V7640–V7646</b>	Reserved	N/A
<b>V7647</b>	Timed Interrupt	Default: 0000 Range: 0003–03E7h (3–9999ms)
<b>V7650–V7654</b>	Reserved	N/A
<b>V7655</b>	Port 2: Setup for the protocol, time-out, and the response delay time.	Default: 00E0
<b>V7656</b>	Port 2: Setup for the station number, baud rate, STOP bit, and parity.	Default: 8501



System V-memory	Description of Contents	Default Values/Ranges
V7657	Port 2: Setup completion code used to notify the completion of the parameter setup	Default: 0A00
V7660	Scan control setup: Keeps the scan control mode	Default: 0000
V7661	Setup timer over counter: Counts the times the actual scan time exceeds the user setup time	N/A
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–V7750	Reserved	
V7751	Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed	
V7752–V7754	Reserved	
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 — 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 optional Memory Cartridge clock (00-59)	
V7767	Contains the number of minutes on optional Memory Cartridge clock (00-59)	
V7770	Contains the number of hours on optional Memory Cartridge clock (00-23)	
V7771	Contains the day of week on optional Memory Cartridge (Mon., Tues., Wed., etc.)	
V7772	Contains the numerical day of month on optional Memory Cartridge (01, 02, etc.)	
V7773	Contains the numerical month on optional Memory Cartridge (01 to 12)	
V7774	Contains the year on optional Memory Cartridge (00 to 99)	
V7775	Scan — stores the current scan time (milliseconds)	
V7776	Scan — stores the minimum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds)	
V7777	Scan — stores the maximum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds)	

## DL05 Memory Map Table

Memory Type	Discrete Memory Reference (octal)	Word Memory Reference (octal)	Decimal	Symbol
Input Points (See note)	X0 – X377	V40400 – V40417	256	X0 
Output Points (See note)	Y0 – Y377	V40500 – V40517	256	Y0 -( )-
Control Relays	C0 – C777	V40600 – V40637	512	C0      C0          -( )-
Special Relays	SP0 – SP777	V41200 – V41237	512	SP0 
Timers	T0 – T177	V41100 – V41107	128	
Timer Current Values	None	V0 – V177	128	V0 K100 - ≥
Timer Status Bits	T0 – T177	V41100 – V41107	128	T0 
Counters	CT0 – CT177	V41140 – V41147	128	
Counter Current Values	None	V1000 – V1177	128	V1000 K100 - ≥
Counter Status Bits	CT0 – CT177	V41140 – V41147	128	CT0 
Data Words (See Appendix F)	None	V1200 – V7377	3968	None specific, used with many instructions.
Data Words Non-volatile (See Appendix F)	None	V7400 – V7577	128	None specific, used with many instructions. May be non-volatile if MOV inst. is used. Data can be rewritten to EEPROM at least 100,000 times before it fails.
Stages	S0 – S377	V41000 – V41017	256	 SP0 
System parameters	None	V7600 – V7777	128	None specific, used for various purposes



**NOTE:** The DL05 has 8 discrete inputs and 6 discrete outputs which are standard. The number of inputs and/or outputs can be increased by adding one of the available option modules. Refer to either the DL05/06 Option Modules User Manual (DO-OPTIONS-M), our catalog or our website.

## DL05 Aliases

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

DL05 Aliases		
Address Start	Alias Start	Example
V0	TA0	V0 is the timer accumulator value for timer 0; therefore, its alias is TA0. TA1 is the alias for V1, etc.
V1000	CTA0	V1000 is the counter accumulator value for counter 0; therefore, its alias is CTA0. CTA1 is the alias for V1001, etc.
V40400	VX0	V40400 is the word memory reference for discrete bits X0 through X17; therefore, its alias is VX0. V40401 is the word memory reference for discrete bits X20 through X37; therefore, its alias is VX20.
V40500	VY0	V40500 is the word memory reference for discrete bits Y0 through Y17; therefore, its alias is VY0. V40501 is the word memory reference for discrete bits Y20 through Y37; therefore, its alias is VY20.
V40600	VC0	V40600 is the word memory reference for discrete bits C0 through C17; therefore, its alias is VC0. V40601 is the word memory reference for discrete bits C20 through C37; therefore, its alias is VC20.
V41000	VS0	V41000 is the word memory reference for discrete bits S0 through S17; therefore, its alias is VS0. V41001 is the word memory reference for discrete bits S20 through S37; therefore, its alias is VS20.
V41100	VT0	V41100 is the word memory reference for discrete bits T0 through T17; therefore, its alias is VT0. V41101 is the word memory reference for discrete bits T20 through T37; therefore, its alias is VT20.
V41140	VCT0	V41140 is the word memory reference for discrete bits CT0 through CT17; therefore, its alias is VCT0. V41141 is the word memory reference for discrete bits CT20 through CT37; therefore, its alias is VCT20.
V41200	VSP0	V41200 is the word memory reference for discrete bits SP0 through SP17; therefore, its alias is VSP0. V41201 is the word memory reference for discrete bits SP20 through SP37; therefore, its alias is VSP20.

## X Input Bit Map

This table provides a listing of individual Input points associated with each V-memory address bit for the DL05's eight physical inputs. Actual available references are X0 to X377 (V40400 – V40417).

3

MSB		DL05 Input (X) Points														Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
–	–	–	–	–	–	–	–	007	006	005	004	003	002	001	000	V40400

This table provides the listing for the individual option slot Input points available.

MSB		DL05 Option Slot Input (X) Points														Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V40404

## Y Output Bit Map

This table provides a listing of individual output points associated with each V-memory address bit for the DL05's six physical outputs. Actual available references are Y0 to Y377 (V40500 – V40517).

MSB		DL05 Output (Y) Points														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
–	–	–	–	–	–	–	–	–	–	005	004	003	002	001	000	V40500	

This table provides the listing for the individual option slot Output points available.

MSB		DL05 Option Slot Output (Y) Points														LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V40504	

## Control Relay Bit Map

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

MSB	DL05 Control Relays (C)															LSB	Address
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
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	

## Stage Control/Status Bit Map

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

3

MSB	DL05 Stage (S) Control Bits															LSB	Address
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	<b>V41000</b>	
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	<b>V41001</b>	
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	<b>V41002</b>	
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	<b>V41003</b>	
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	<b>V41004</b>	
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	<b>V41005</b>	
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	<b>V41006</b>	
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	<b>V41007</b>	
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	<b>V41010</b>	
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	<b>V41011</b>	
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	<b>V41012</b>	
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	<b>V41013</b>	
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	<b>V41014</b>	
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	<b>V41015</b>	
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	<b>V41016</b>	
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	<b>V41017</b>	

## Timer Status Bit Map

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

MSB	DL05 Timer (T) Contacts															LSB	Address
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	<b>V41100</b>	
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	<b>V41101</b>	
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	<b>V41102</b>	
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	<b>V41103</b>	
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	<b>V41104</b>	
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	<b>V41105</b>	
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	<b>V41106</b>	
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	<b>V41107</b>	

## Counter Status Bit Map

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

MSB	DL05 Counter (CT) Contacts															LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	<b>V41140</b>	
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	<b>V41141</b>	
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	<b>V41142</b>	
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	<b>V41143</b>	
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	<b>V41144</b>	
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	<b>V41145</b>	
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	<b>V41146</b>	
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	<b>V41147</b>	

# CONFIGURATION AND CONNECTIONS

---



## In This Chapter:

DL05 System Design Strategies .....	4-2
Network Configuration and Connections .....	4-4
Network Slave Operation .....	4-8
Network Master Operation .....	4-14



## DL05 System Design Strategies

### I/O System Configurations

The DL05 PLCs offer a number of different I/O configurations. Choose the configuration that is right for your application, and keep in mind that the DL05 PLCs offer the ability to add an I/O card in the option slot. Although remote I/O isn't available, there are several option cards available. For instance:

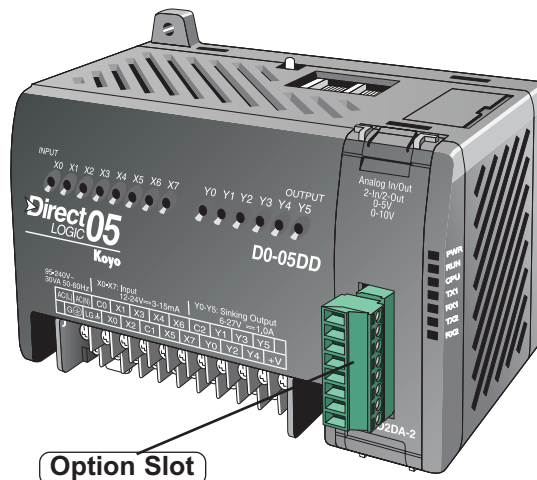
- Various A/C and D/C I/O modules
- Combination I/O modules
- Analog I/O modules
- Combination Analog I/O modules

A DL05 system can be developed with an arrangement using a selected option modules. See our DL05/06 Options Modules User Manual (D0-OPTIONS-M) on the website, [www.automationdirect.com](http://www.automationdirect.com) for detailed selection information.

### Networking Configurations

The DL05 PLCs offers the following ways to add networking:

- **Ethernet Communications Module** – connects a DL05 to high-speed peer-to-peer networks. Any PLC can initiate communications with any other PLC or operator interfaces, such as C-more, when using the ECOM modules.
- **Data Communications Modules** – connects a DL05 to devices using either DeviceNet or Profibus to link to master controllers, as well as a D0-DCM.
- **Communications Port 1** – The DL05 has a 6-pin RJ12 connector on Port 1 that supports (as slave) K-sequence, Modbus RTU or *DirectNET* protocols.
- **Communications Port 2** – The DL05 has a 6-pin RJ12 connector on Port 2 that supports either master/slave Modbus RTU or *DirectNET* protocols, or K-sequence protocol as slave. Port 2 can also be used for ASCII OUT communications.



### Automatic I/O Configuration

The DL05 CPUs will automatically detect the optional I/O module, if installed, at powerup and establish the correct I/O configuration and addresses. The configuration may never need to be changed.

The I/O addresses use octal numbering, with X0 to X7 being the eight inputs and Y0 to Y5 being the addresses for the six outputs. The discrete option slot addresses are assigned in groups of 8 or 16 depending on the number of I/O points for the I/O module. The discrete option module addressing will be X100 to X107 and X110 to X117 for the maximum sixteen point input module. The addressing for the sixteen point output module will be Y100 to Y107 and Y110 to Y117. Refer to the DL05/06 Options Modules User Manual (D0-OPTIONS-M) for the various discrete I/O modules available and the addressing for each one.

### Power Budgeting

No power budgeting is necessary for the DL05. The built-in power supply is sufficient for powering the base unit, your choice of option module, the handheld programmer and the DV-1000 operator interface.

# Network Configuration and Connections

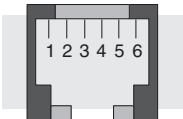
## Configuring the DL05's Comm Ports

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 DL05 PLC system directly to Modbus networks using the RTU protocol, or to other devices on a *DirectNET* network. Modbus host systems must be capable of issuing the Modbus commands to read or write the appropriate data. For details on the Modbus protocol, check with your Modbus supplier for the latest version of the Gould Modbus Protocol reference Guide. For more details on *DirectNET*, order our *DirectNET* manual, part number DA-DNET-M.

### DL05 Port Specifications

Communications Port 1		Communications Port 2	
Port 1	Connects to HPP, <i>DirectSOFT</i> , operator interfaces, etc.	Port 2	Connects to HPP, <i>DirectSOFT</i> , operator interfaces, etc.
	6-pin, RS232C		6-pin, multifunction port, RS232C
	Communication speed: 9600 Baud (fixed)		Communication speed (baud): 300, 600, 1200, 2400, 4800, 9600, 19200, 38400
	Parity: odd (fixed)		Parity: odd (default), even, none
	Station Address: 1 (fixed)		Station Address: 1 (default)
	8 data bits		8 data bits
	1 start, 1 stop bit		1 start, 1 stop bit
	Asynchronous, half-duplex, DTE		Asynchronous, half-duplex, DTE
	Protocol (auto-select): K-sequence (slave only), <i>DirectNET</i> (slave only), Modbus RTU (slave only)		Protocol (auto-select): K-sequence (slave only), <i>DirectNET</i> (master/slave), Modbus RTU (master/slave), non-sequence/print

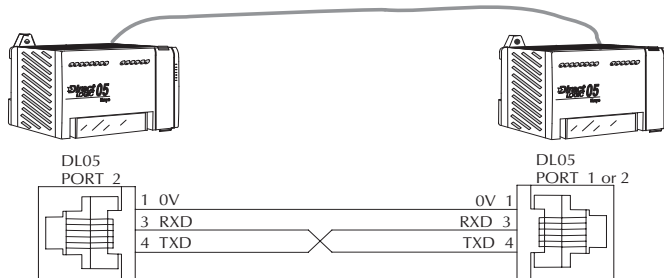
### Networking



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

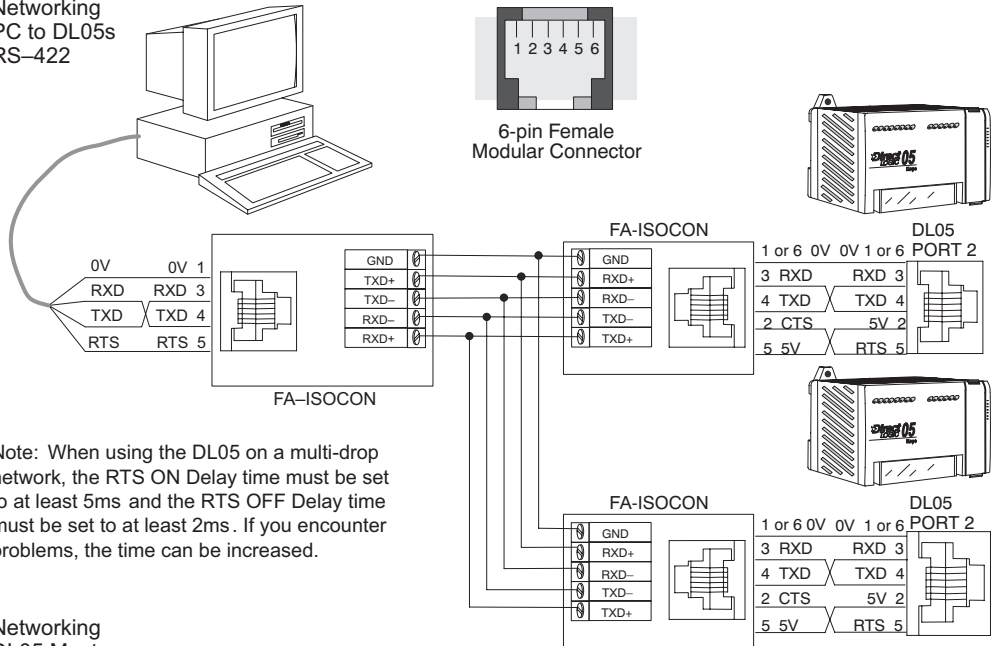
### DL05 to DL05 RS-232C

You will need to make sure the network connection is a 3-wire RS-232 type. The recommended cable is AutomationDirect L19772 (Belden 8102) or equivalent. Normally, the RS-232 signals are used for communications between two devices with distances up to a maximum of 15 meters.



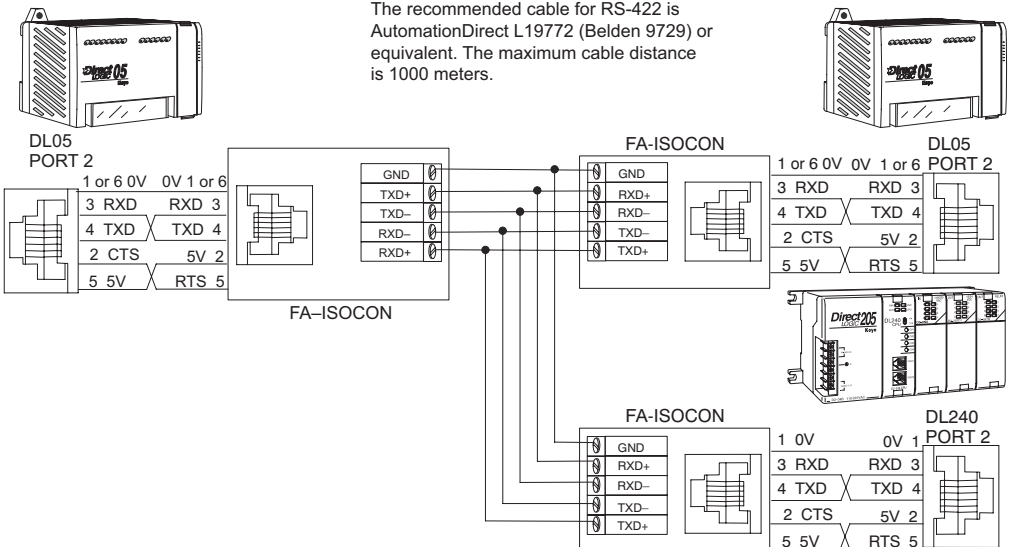
## Networking Using RS-422 Converters

Networking  
PC to DL05s  
RS-422



Note: When using the DL05 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.

Networking  
DL05 Master  
to Other PLCs

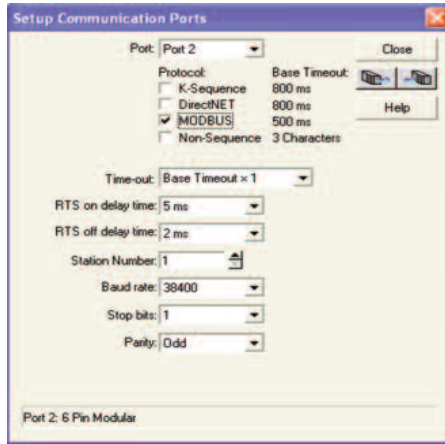


The recommended cable for RS-422 is AutomationDirect L19772 (Belden 9729) or equivalent. The maximum cable distance is 1000 meters.

### Modbus Port Configuration

In *DirectSOFT 5*, 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 DL05 waits to send the data after it has raised the RTS signal line. The RTS OFF Delay Time specifies the time the DL05 waits to release the RTS signal line after the data has been sent. *When using the DL05 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:** The possible range for Modbus slave numbers is from 1 to 247, but the DL05 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 DL05 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.

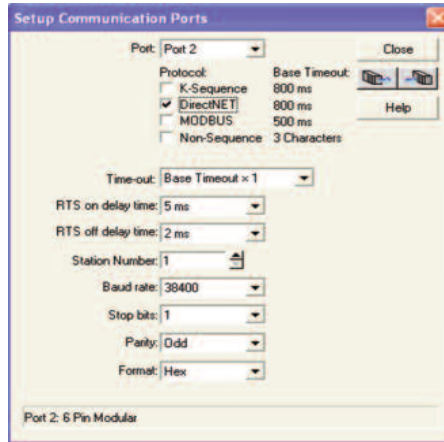


Then click the button indicated to send the Port configuration to the CPU, and click Close.

## DirectNET Port Configuration

In *DirectSOFT 5*, 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 DL05 waits to send the data after it has raised the RTS signal line. The RTS OFF Delay Time specifies the time the DL05 waits to release the RTS signal line after the data has been sent. *When using the DL05 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 DL05 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.

## 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 (DL05). A Modbus host must use the Modbus RTU protocol to communicate with the DL05 as a slave. The host software must send a Modbus function code and Modbus address to specify a PLC memory location the DL05 comprehends. The *DirectNET* host uses normal I/O addresses to access applicable DL05 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 DL05 supports the Modbus function codes described below.

MODBUS Function Code	Function	DL05 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	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	V
16	Write a value into a group of registers	V

### Determining the Modbus Address

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

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



**NOTE:** For information about the Modbus protocol see the Group Schneider website at: [www.schneiderautomation.com](http://www.schneiderautomation.com). At the main menu, select Support/Services, Modbus, Modbus Technical Manuals, PI-MBUS-300 Modbus Protocol Reference Guide or search for PIMBUS300. For more information about the *DirectNET* protocol, order our *DirectNET* User Manual, DA-DNET-M, or download the manual free from our website: [www.automationdirect.com](http://www.automationdirect.com). Select Manuals\Docs\onlineusermanuals\misc\DA-DNET-M

## 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.

DL05 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>				
Inputs (X)	256	X0 – X377	2048 – 2303	Input
Special Relays(SP)	512	SP0 – SP777	3072 – 3583	Input
Outputs (Y)	256	Y0 – Y377	2048 – 2303	Coil
Control Relays (CR)	512	C0 – C777	3072 – 4583	Coil
Timer Contacts (T)	128	T0 – T177	6144 – 6271	Coil
Counter Contacts (CT)	128	CT0 – CT177	6400 – 6527	Coil
Stage Status Bits(S)	256	S0 – S377	5120 – 5375	Coil
<b>For Word Data Types .... Convert PLC Addr. to Dec. + Data Type</b>				
Timer Current Values (V)	128	V0 – V177	0 – 127	Input Register
Counter Current Values (V)	128	V1000 – V1177	512 – 639	Input Register
V-Memory, user data (V)	3968	V1200 – V7377	640 – 3839	Holding Register
V-Memory, non-volatile (V)	128	V7600 – V7777	3968 – 4095	Holding Register



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

## Example 1: V2100

Find the Modbus address for User V location V2100.

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

**PLC Address (Dec) + Data Type**

V2100 = 1088 decimal

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

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

## Example 2: Y20

Find the Modbus address for output Y20.

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

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

Y20 = 16 decimal

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

## Example 3: T10 Current Value

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

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

**PLC Address (Dec) + Data Type**

T10 = 8 decimal

8 + Input Reg. = **Input Reg. 8**

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

## Example 4: C54

Find the Modbus address for Control Relay C54.

1. Find Control Relays in the table.
2. Convert C54 into decimal (44).
3. Add the starting address for the range (3072).
4. Use the Modbus data type from the table.

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

C54 = 44 decimal

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

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

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 (contacts), C (contacts)
- Word – V, Timer current value, Counter current value

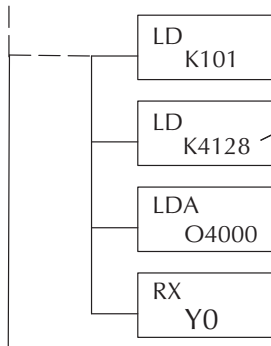
In either case, you basically convert the PLC octal address to decimal and add the appropriate Modbus addresses (as required). The table below shows the exact equation used for each group of data.

Discrete Data Types				
DL05 Memory Type	PLC Range (Octal)	Address (484 Mode)	Address (584/984 Mode)	Modbus Data Type
Global Inputs (GX)	GX0-GX1746	1001 - 1999	10001 - 10999	Input
	GX1747-GX3777	---	11000 - 12048	Input
Inputs (X)	X0 - X1777	---	12049 - 13072	Input
Special Relays (SP)	SP0 - SP777	---	13073 - 13584	Input
Global Outputs (GY)	GY0 - GY3777	1 - 2048	1 - 2048	Output
Outputs (Y)	Y0 - Y1777	2049 - 3072	2049 - 3072	Output
Control Relays (CR)	C0 - C3777	3073 - 5120	3073 - 5120	Output
Timer Contacts (T)	T0 - T377	6145 - 6400	6145 - 6400	Output
Counter Contacts (CT)	CT0 - CT377	6401 - 6656	6401 - 6656	Output
Stage Status Bits (S)	S0 - S1777	5121 - 6144	5121 - 6144	Output

Word Data Types			
Registers	PLC Range (Octal)	Input/Holding (484 Mode)*	Input/Holding (584/984 Mode)*
V-Memory (Timers)	V0 - V377	3001/4001	30001/40001
V-Memory (Counters)	V1000 - V1177	3513/4513	30513/40513
V-Memory (Data Words)	V1200 - V1377	3641/4641	30641/40641
	V1400 - V1746	3769/4769	30769/40769
	V1747 - V1777	---	31000/41000
	V2000 - V7377	---	41025
	V10000 - V17777	---	44097

\* Modbus: Function 04

The DL05/06, DL250-1/260, DL350 and DL450 will support function 04, read input register (Address 30001). To use function 04, put the number '4' into the most significant position (4xxx). Four digits must be entered for the instruction to work properly with this mode.



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

1. Refer to your PLC user manual for the correct memory mapping size of your PLC. Some of the addresses shown above might not pertain to your particular CPU.
2. For an automated Modbus/Koyo address conversion utility, download the file [modbus\\_conversion.xls](http://www.automationdirect.com) from the [www.automationdirect.com](http://www.automationdirect.com) website.

### Example 1: V2100 584/984 Mode

Find the Modbus address for user V-memory V2100. PLC Address (Dec) + Mode Address

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

$$V2100 = 1088 \text{ decimal}$$

$$1088 + 40001 = \boxed{41089}$$

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

### Example 2: Y20 584/984 Mode

Find the Modbus address for output Y20.

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

$$\text{PLC Address (Dec) + Start Addr + Mode}$$

$$Y20 = 16 \text{ decimal}$$

$$16 + 2048 + 1 = \boxed{2065}$$

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

### Example 3: T10 Current Value 484 Mode

Find the Modbus address to obtain the current value for Timer T10. PLC Address (Dec) + Mode Address

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

$$T10 = 8 \text{ decimal}$$

$$8 + 3001 = \boxed{3009}$$

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

### Example 4: C54 584/984 Mode

Find the Modbus address for Control Relay C54.

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

$$\text{PLC Address (Dec) + Start Addr + Mode}$$

$$C54 = 44 \text{ decimal}$$

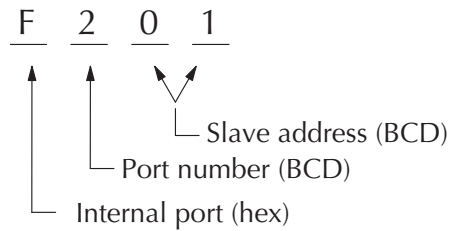
$$44 + 3072 + 1 = \boxed{3117}$$

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



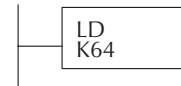
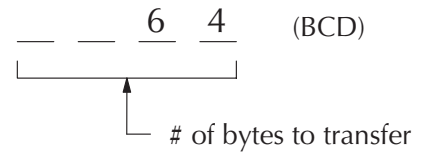
**Step 1: Identify Master Port # and Slave #**

The first Load (LD) instruction identifies the communications port number on the network master (DL05) and the address of the slave station. This instruction can address up to 99 Modbus slaves, or 90 *DirectNET* slaves. The format of the word is shown to the right. The “F2” in the upper byte indicates the use of the right port of the DL05 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 DL05 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 *DirectLOGIC™* products.

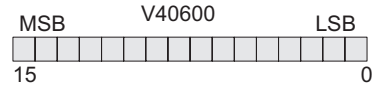
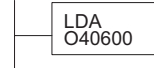
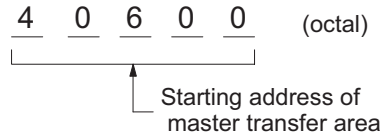
DL 05/205/350/405 Memory	Bits per unit	Bytes
V-memory	16	2
T / C current value	16	2
Inputs (X, SP)	8	1
Outputs (Y, C, Stage, T/C bits)	8	1
Scratch Pad Memory	8	1
Diagnostic Status	8	1
DL330/340 Memory	Bits per unit	Bytes
Data registers	8	1
T / C accumulator	16	2
I/O, internal relays, shift register bits, 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 DL05 CPU sends the number of bytes previously specified from its memory area beginning at the LDA address specified.

For an RX instruction, the DL05 CPU reads the number of bytes previously specified from the slave, placing the received data into its memory area beginning at the LDA address specified.



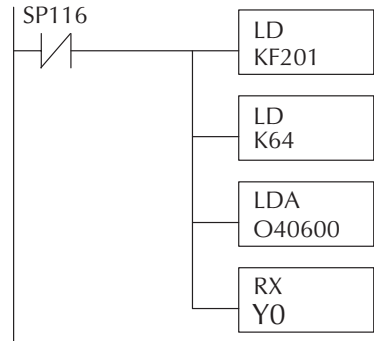
**NOTE:** Since V-memory words are always 16 bits, you may not always use the whole word. For example, if you only specify 3 bytes and you are reading Y outputs from the slave, you will only get 24 bits of data. In this case, only the 8 least significant bits of the last word location will be modified.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



### Step 4: Specify Slave Memory Area

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

- *Direct*NET slaves – specify the same address in the WX and RX instruction as the slave’s native I/O address
- Modbus DL405, DL205, or DL05 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



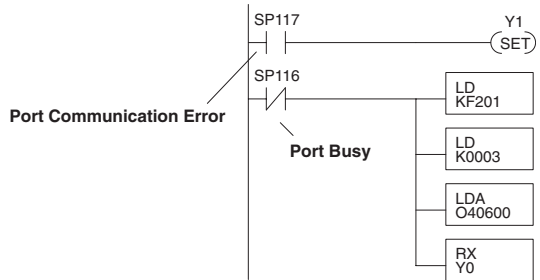
**DL305 Series CPU Memory Type-to-Modbus Cross Reference (excluding 350 CPU)**

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.

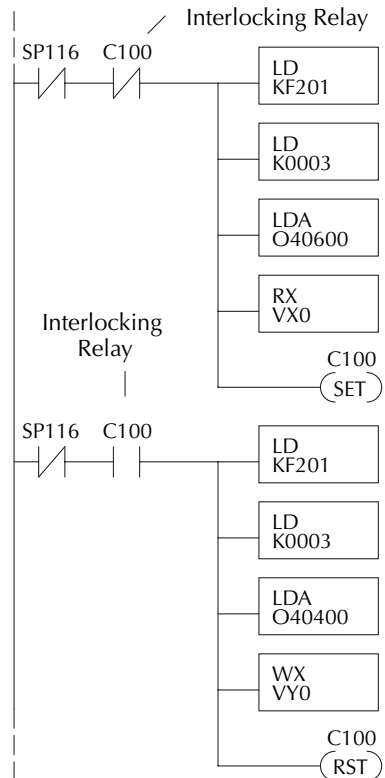


The “Port Communication Error” bit turns on when the PLC has detected an error. Use of this bit is optional. When used, it should be ahead of any network instruction boxes since the error bit is reset when an RX or WX instruction is executed.

## 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.





# STANDARD RLL AND INTELLIGENT BOX INSTRUCTIONS

---



## In This Chapter:

Introduction .....	.5-2
Using Boolean Instructions .....	.5-4
Boolean Instructions .....	.5-9
Comparative Boolean .....	.5-25
Immediate Instructions .....	.5-31
Timer, Counter and Shift Register Instructions .....	.5-35
Accumulator/Stack Load and Output Data Instructions .....	.5-48
Logical Instructions (Accumulator) .....	.5-60
Math Instructions .....	.5-68
Bit Operation Instructions .....	.5-82
Number Conversion Instructions (Accumulator) .....	.5-87
Table Instructions .....	.5-96
CPU Control Instructions .....	.5-99
Program Control Instructions .....	.5-101
Interrupt Instructions .....	.5-108
Message Instructions .....	.5-111
Intelligent I/O Instructions .....	.5-118
Network Instructions .....	.5-120
Intelligent Box (IBox) Instructions .....	.5-124

# Introduction

DL05 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.

5

Instruction	Page	Instruction	Page
Accumulating Timer (TMRA)	5-38	Decode (DECO)	5-86
Accumulating Fast Timer (TMRAF)	5-38	Decrement (DEC)	5-76
Add (ADD)	5-68	Decrement Binary (DECB)	5-77
Add Binary (ADDB)	5-78	Disable Interrupts (DISI)	5-109
Add Double (ADDD)	5-69	Divide (DIV)	5-74
And (AND)	5-13	Divide Binary (DIVB)	5-81
And (AND)	5-30	Divide Double (DIVD)	5-75
And (AND)	5-60	Enable Interrupts (ENI)	5-108
And Bit-of-Word (ANDB)	5-14	Encode (ENCO)	5-85
And Double (ANDD)	5-61	End (END)	5-99
And If Equal (ANDE)	5-27	Exclusive Or (XOR)	5-64
And If Not Equal (ANDNE)	5-27	Exclusive Or Double (XORD)	5-65
And Immediate (ANDI)	5-32	Fault (FAULT)	5-111
And Negative Differential (ANDND)	5-21	For / Next (FOR) (NEXT)	5-101
And Not (ANDN)	5-13	Goto Subroutine (GTS) (SBR)	5-103
And Not (ANDN)	5-30	Gray Code (GRAY)	5-93
And Not Bit-of-Word (ANDNB)	5-14	HEX to ASCII (HTA)	5-91
And Not Immediate (ANDNI)	5-32	Increment (INC)	5-76
And Positive Differential (ANDPD)	5-21	Increment Binary (INCB)	5-77
And Store (AND STR)	5-15	Interrupt (INT)	5-108
ASCII Constant (ACON)	5-112	Interrupt Return (IRT)	5-108
ASCII to HEX (ATH)	5-90	Interrupt Return Conditional (IRTC)	5-108
Binary (BIN)	5-87	Invert (INV)	5-89
Binary Coded Decimal (BCD)	5-88	Load (LD)	5-53
Compare (CMP)	5-66	Load Address (LDA)	5-56
Compare Double (CMPD)	5-67	Load Double (LDD)	5-54
Counter (CNT)	5-41	Load Formatted (LDF)	5-55
Data Label (DLBL)	5-112	Load Label (LDLBL)	5-97

Instruction	Page	Instruction	Page
Master Line Reset (MLR)	5-106	Reset Bit-of-Word (RSTB)	5-23
Master Line Set (MLS)	5-106	Reset Immediate (RSTI)	5-34
Move (MOV)	5-96	Reset Watch Dog Timer (RSTWT)	5-100
Move Memory Cartridge (MOVMC)	5-97	Set (SET)	5-22
Multiply (MUL)	5-72	Set Bit-of-Word (SETB)	5-23
Multiply Binary (MULB)	5-80	Set Immediate (SETI)	5-34
Multiply Double (MULD)	5-73	Shift Left (SHFL)	5-83
No Operation (NOP)	5-99	Shift Register (SR)	5-47
Not (NOT)	5-18	Shift Right (SHFR)	5-84
Numerical Constant (NCON)	5-112	Shuffle Digits (SFLDGT)	5-94
Or (OR)	5-11	Stage Counter (SGCNT)	5-43
Or (OR)	5-29	Stop (STOP)	5-99
Or (OR)	5-62	Store (STR)	5-9
Or Bit-of-Word (ORB)	5-12	Store (STR)	5-28
Or Double (ORD)	5-63	Store Bit-of-Word (STRB)	5-10
Or If Equal (ORE)	5-26	Store If Equal (STRE)	5-25
Or If Not Equal (ORNE)	5-26	Store If Not Equal (STRNE)	5-25
Or Immediate (ORI)	5-31	Store Immediate (STRI)	5-31
Or Negative Differential (ORND)	5-20	Store Negative Differential (STRND)	5-19
Or Not (ORN)	5-11	Store Not (STRN)	5-9
Or Not (ORN)	5-29	Store Not (STRN)	5-28
Or Not Bit-of-Word (ORNB)	5-12	Store Not Bit-of-Word (STRNB)	5-10
Or Not Immediate (ORNI)	5-31	Store Not Immediate (STRNI)	5-31
Or Out (OR OUT)	5-16	Store Positive Differential (STRPD)	5-19
Or Out Immediate (OROUTI)	5-33	Subroutine Return (RT)	5-103
Or Positive Differential (ORPD)	5-20	Subroutine Return Conditional (RTC)	5-103
Or Store (OR STR)	5-15	Subtract (SUB)	5-70
Out (OUT)	5-16	Subtract Binary (SUBB)	5-79
Out (OUT)	5-57	Subtract Double (SUBD)	5-71
Out Bit-of-Word (OUTB)	5-17	Sum (SUM)	5-81
Out Double (OUTD)	5-57	Timer (TMR) and Timer Fast (TMRF)	5-36
Out Formatted (OUTF)	5-58	Up Down Counter (UDC)	5-45
Out Immediate (OUTI)	5-33	Write to Intelligent Box I/O Module (WT)	5-119
Pause (PAUSE)	5-24	Write to Network (WX)	5-122
Pop (POP)	5-58		
Positive Differential (PD)	5-18		
Print Message (PRINT)	5-114		
Read from Intelligent Box I/O Module (RD)	5-118		
Read from Network (RX)	5-120		
Reset (RST)	5-22		

## 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. Our *DirectSOFT 5* software is a similar program. It uses graphic symbols to develop a program; therefore, you don't necessarily have to know the instruction mnemonics in order to develop your program. However, knowledge of mnemonics will be helpful, whenever it becomes necessary to troubleshoot a program using a handheld programmer (HPP).

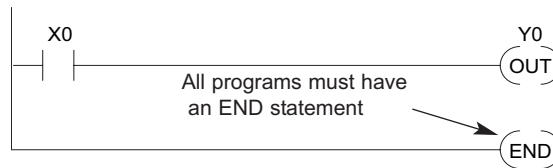
Many of the instructions in this chapter are not program instructions used in *DirectSOFT 5*, but are implied. In other words, they are not actually keyboard commands, however, they can be seen in a Mnemonic View of the program once the *DirectSOFT 5* program has been developed and accepted (compiled). Each instruction listed in this chapter will have a small chart to indicate how the instruction is used with *DirectSOFT 5* and the HPP.

DS5	Implied
HPP	Used

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

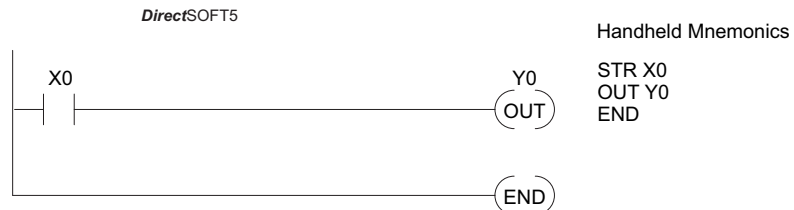
### END Statement

All DL05 programs require an END statement as the last instruction. This tells the CPU that this is the end of the program. Normally, any instructions placed after the END statement will not be executed. There are exceptions to this such as interrupt routines, etc.. This chapter will discuss the instruction set in detail.



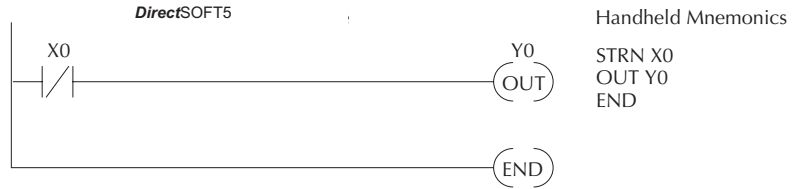
### Simple Rungs

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



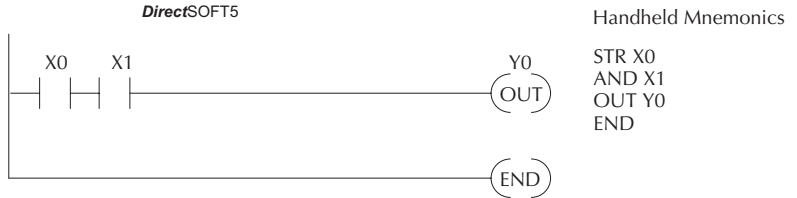
### Normally Closed Contact

Normally closed contacts are also very common. This is accomplished with the Store Not or, STRN instruction. The following example shows a simple rung with a normally closed contact.



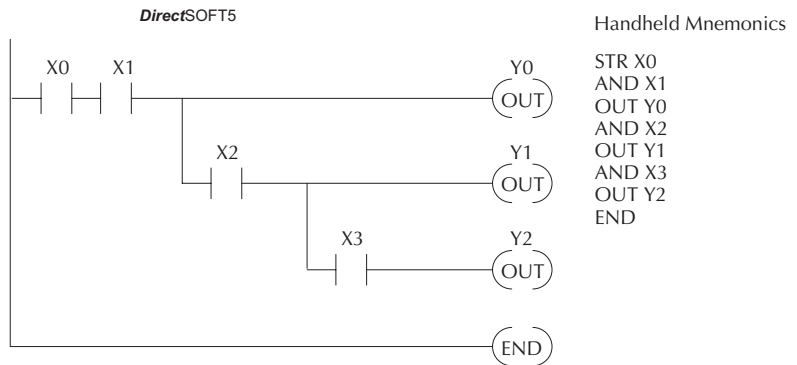
### Contacts in Series

Use the AND instruction to join two or more contacts in series. The following example shows two contacts in series and a single output coil. The instructions used would be STR X0, AND X1, followed by OUT Y0.



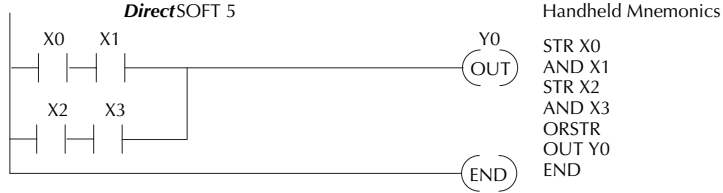
### Midline Outputs

Sometimes it is necessary to use midline outputs to get additional outputs that are conditional on other contacts. The following example shows how you can use the AND instruction to continue a rung with more conditional outputs.



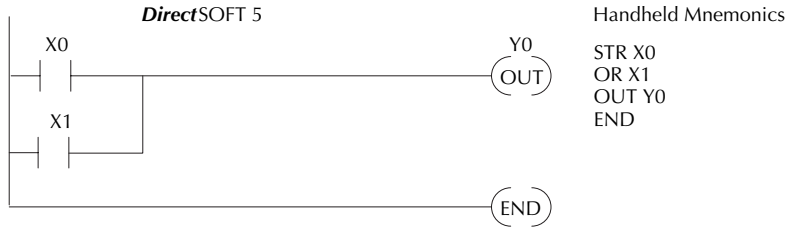
### Parallel Elements

You 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.



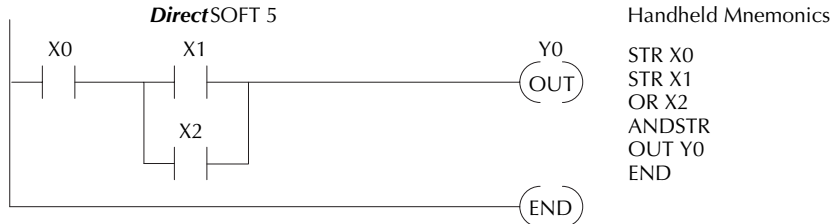
### Joining Series Branches in Parallel

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.



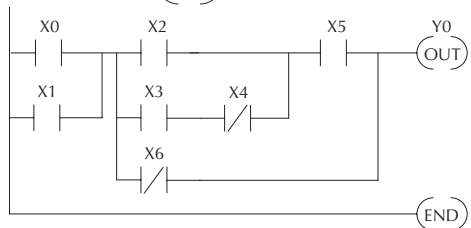
### Joining Parallel Branches in Series

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.



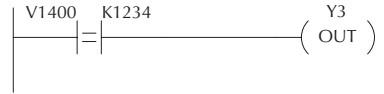
### Combination Networks

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 DL05 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.

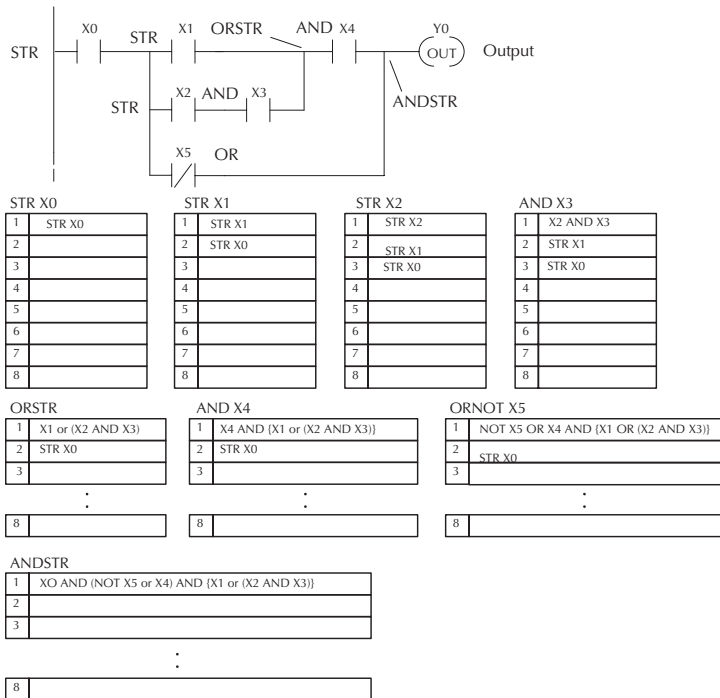


In the following example when the value in V-memory location V1400 is equal to the constant value 1234, Y3 will energize.

### Boolean Stack

There are limits to how many elements you can include in a rung. This is because the DL05 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.

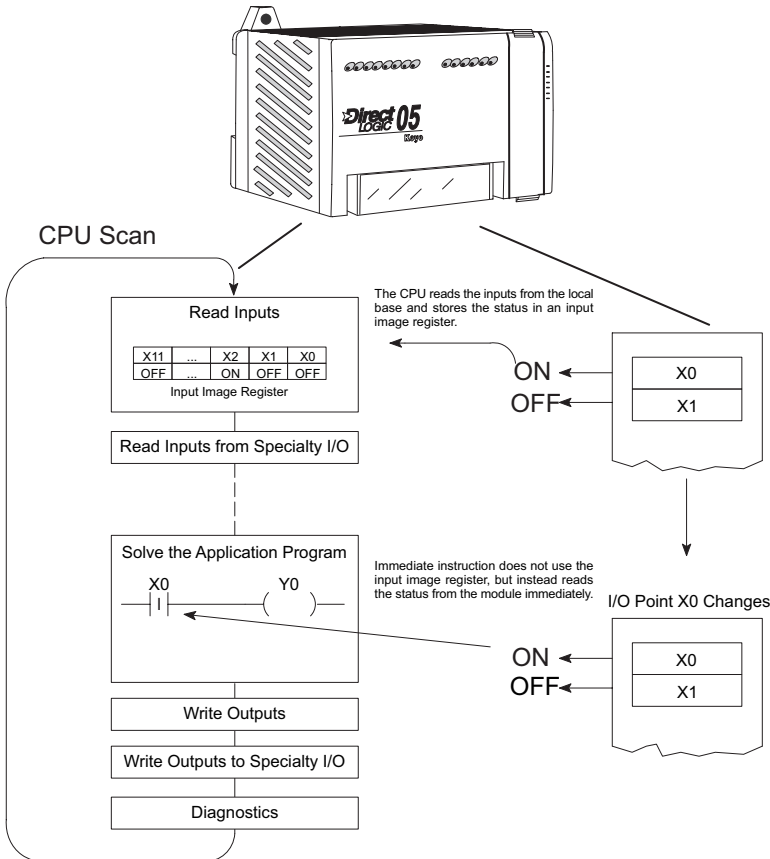


### Immediate Boolean

The DL05 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 DL05 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.

5

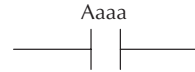




# Boolean Instructions

## Store (STR)

DS5	Implied	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.
HPP	Used	



## Store Not (STRN)

DS5	Implied	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.
HPP	Used	



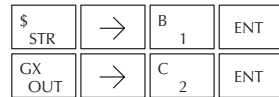
Operand Data Type	DL05 Range
..... A	aaa
Inputs ..... X	0-377
Outputs ..... Y	0-377
Control Relays ..... C	0-777
Stage ..... S	0-377
Timer ..... T	0-177
Counter C ..... T	0-177
Special Relay ..... SP	0-777

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

DirectSOFT 5



Handheld Programmer Keystrokes

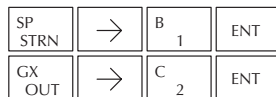


In the following Store Not example, when input X1 is off output Y2 will energize.

DirectSOFT 5



Handheld Programmer Keystrokes



### Store Bit-of-Word (STRB)

DS5	Implied
HPP	Used

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)

DS5	Implied
HPP	Used

The Store Not Bit-of-Word 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.



5

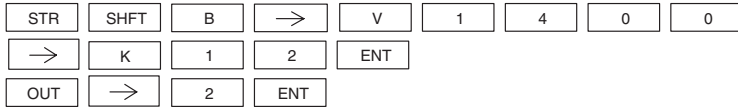
Operand Data Type		DL05 Range	
.....	A	aaa	bb
V-memory .....	B	See memory map	BCD, 0 to 15
Pointer .....	PB	See memory map	BCD, 0 to 15

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

DirectSOFT 5



Handheld Programmer Keystrokes

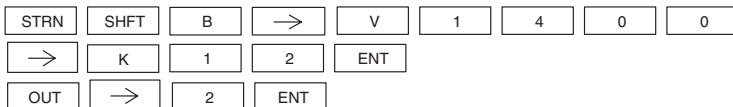


In the following Store Not Bit-of-Word example, when bit 12 of V-memory location V1400 is off, output Y2 will energize.

DirectSOFT 5



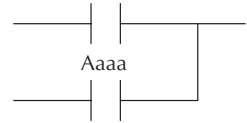
Handheld Programmer Keystrokes



**Or (OR)**

DS5	Implied
HPP	Used

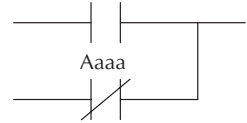
The Or instruction logically ors a normally open contact in parallel with another contact in a rung. The status of the contact will be the same state as the associated image register point or memory location.



**Or Not (ORN)**

DS5	Implied
HPP	Used

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	DL05 Range
..... A	aaa
Inputs ..... X	0-377
Outputs ..... Y	0-377
Control Relays ..... C	0-777
Stage ..... S	0-377
Timer ..... T	0-177
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.

DirectSOFT 5



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
Q OR	→	C 2	ENT
GX OUT	→	F 5	ENT

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

DirectSOFT 5



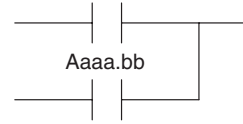
Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
R ORN	→	C 2	ENT
GX OUT	→	F 5	ENT

### Or Bit-of-Word (ORB)

DS5	Implied
HPP	Used

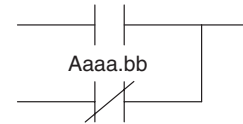
The Or Bit-of-Word instruction logically ors a normally open contact in parallel with another contact in a rung. Status of the contact will be the same state as the bit referenced in the associated memory location.



### Or Not Bit-of-Word (ORNB)

DS5	Implied
HPP	Used

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.

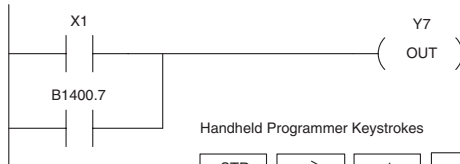


5

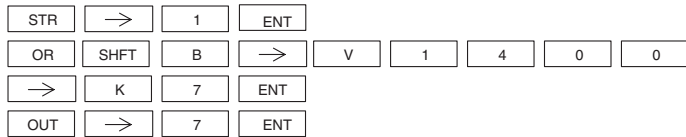
Operand Data Type		DL05 Range	
.....	A	<b>aaa</b>	<b>bb</b>
V-memory.....	B	See memory map	BCD, 0 to 15
Pointer .....	PB	See memory map	BCD, 0 to 15

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

DirectSOFT 5

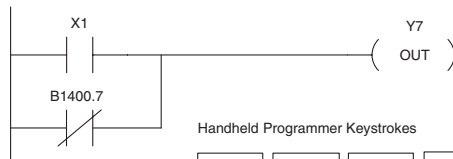


Handheld Programmer Keystrokes

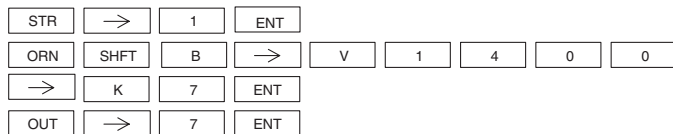


In the following Or Bit-of-Word example, when input X1 is on or bit 7 of V1400 is off, output Y7 will energize.

DirectSOFT 5



Handheld Programmer Keystrokes



### And (AND)

DS5	Implied
HPP	Used

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)

DS5	Implied
HPP	Used

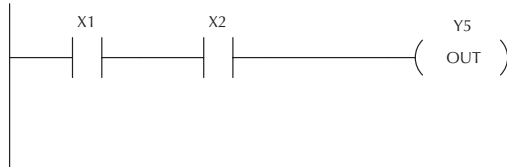
The And Not instruction logically ands a normally closed contact in series with another contact in a rung. The status of the contact will be opposite the state of the associated image register point or memory location.



Operand Data Type	DL05 Range
..... A	aaa
Inputs ..... X	0-377
Outputs ..... Y	0-377
Control Relays ..... C	0-777
Stage ..... S	0-377
Timer ..... T	0-177
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.

DirectSOFT 5

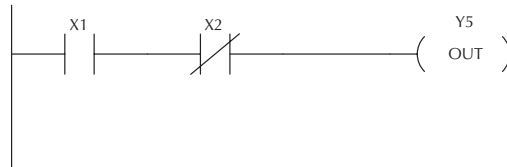


Handheld Programmer Keystrokes

\$ 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.

DirectSOFT 5



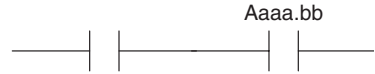
Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
W ANDN	→	C 2	ENT
GX OUT	→	F 5	ENT

### And Bit-of-Word (ANDB)

DS5	Implied
HPP	Used

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)

DS5	Implied
HPP	Used

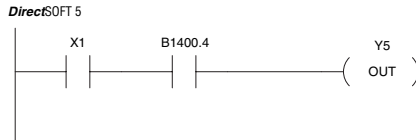
The And Not Bit-of-Word instruction logically ands a normally closed contact in series with another contact in a rung. The status of the contact will be opposite the state of the bit referenced in the associated memory location.



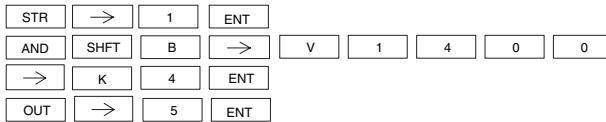
5

Operand Data Type		DL05 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

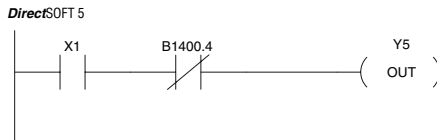
In the following And Bit-of-Word example, when input X1 and bit 4 of V1400 is on output Y5 will energize.



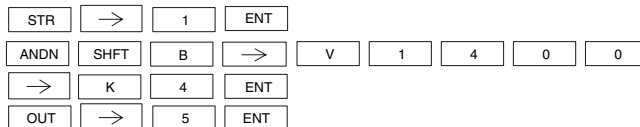
Handheld Programmer Keystrokes



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.



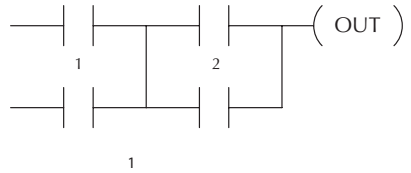
Handheld Programmer Keystrokes



### And Store (AND STR)

DS5	Implied
HPP	Used

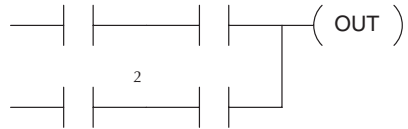
The And Store instruction logically ands two branches of a rung in series. Both branches must begin with the Store instruction.



### Or Store (OR STR)

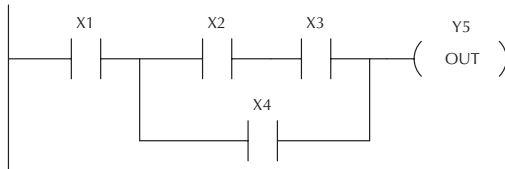
DS5	Implied
HPP	Used

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



In the following And Store example, the branch consisting of contacts X2, X3, and X4 have been anded with the branch consisting of contact X1.

DirectSOFT 5

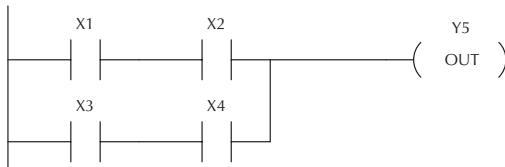


Handheld Programmer Keystrokes

\$ 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.

DirectSOFT 5



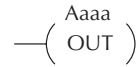
Handheld Programmer Keystrokes

\$ 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)

DS5	Used
HPP	Used

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	DL05 Range
..... A	aaa
Inputs ..... X	0-377
Outputs ..... Y	0-377
Control Relays ..... C	0-777

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

DirectSOFT 5



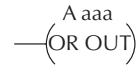
Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
GX OUT	→	C 2	ENT
GX OUT	→	F 5	ENT

### Or Out (OROUT)

DS5	Used
HPP	Used

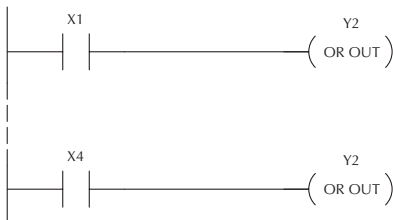
The Or Out instruction allows more than one rung of discrete logic to control a single output. Multiple Or Out instructions referencing the same output coil may be used, since *all* contacts controlling the output are logically ORed together. If the status of *any* rung is on, the output will also be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



Operand Data Type	DL05 Range
..... A	aaa
Inputs ..... X	0-177
Outputs ..... Y	0-177
Control Relays ..... C	0-777

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

DirectSOFT 5



Handheld Programmer Keystrokes

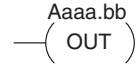
\$ STR	→	B 1	ENT				
O INST#	D 3	F 5	ENT	ENT	→	C 2	ENT
\$ STR	→	E 4	ENT				
O INST#	D 3	F 5	ENT	ENT	→	C 2	ENT



### Out Bit-of-Word (OUTB)

DS5	Used
HPP	Used

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.

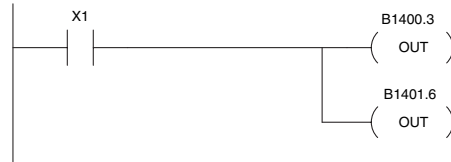


Operand Data Type		DL05 Range	
..... A		aaa	bb
V-memory .....	B	See memory map	BCD, 0 to 15
Pointer .....	PB	See memory map	BCD, 0 to 15

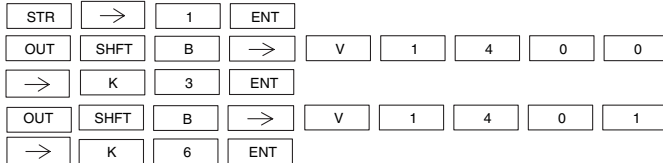
5

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

DirectSOFT 5

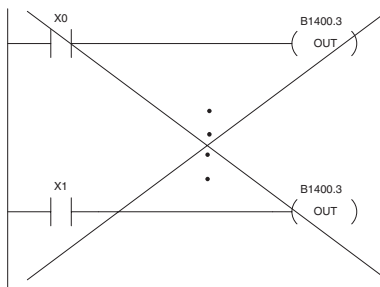


Handheld Programmer Keystrokes



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.

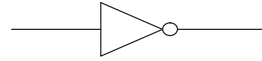
location must not be used in programming.



### Not (NOT)

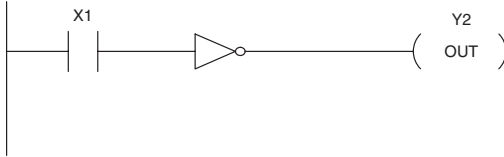
DS5	Used
HPP	Used

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



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

DirectSOFT 5



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
SHFT	N TMR	O INST#	T MLR ENT
GX OUT	→	C 2	ENT

5

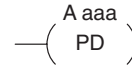


**NOTE:** DirectSOFT Release 1.1i and later supports the use of the NOT instruction. The above example rung is merely intended to show the visual representation of the NOT instruction. The rung cannot be created or displayed in DirectSOFT versions earlier than 1.1i.

### Positive Differential (PD)

DS5	Used
HPP	Used

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	DL05 Range
..... A	aaa
Inputs ..... X	0-377
Outputs ..... Y	0-377
Control Relays ..... C	0-777

In the following example, every time X1 makes an off to on transition, C0 will energize for one scan.

DirectSOFT 5



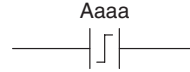
Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
SHFT	P CV	SHFT	D 3 → A 0

### Store Positive Differential (STRPD)

DS5	Used
HPP	Used

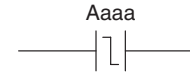
The Store Positive Differential instruction begins a new rung or an additional branch in a rung with a normally open contact. The contact closes for one CPU scan when the state of the associated image register point makes an Off-to-On transition. Thereafter, the contact remains open until the next Off-to-On transition (the symbol inside the contact represents the transition). This function is sometimes called a “one-shot”. This contact will also close on a program-to-run transition if it is within a retentative range and on before the PLC mode transition.



DS5	Used
HPP	Used

### Store Negative Differential (STRND)

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



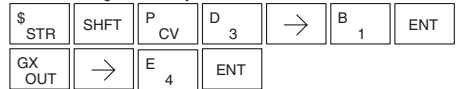
Operand Data Type	DL05 Range
..... A	aaa
Inputs ..... X	0-377
Outputs ..... Y	0-377
Control Relays ..... C	0-777
Stage ..... S	0-377
Timer ..... T	0-177
Counter ..... CT	0-177

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

DirectSOFT 5



Handheld Programmer Keystrokes

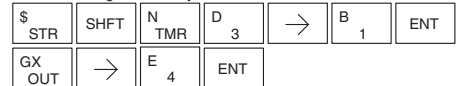


In the following example, each time X1 is makes an On-to-Off transition, Y4 will energize for one scan.

DirectSOFT 5



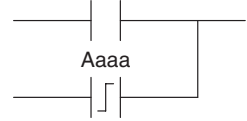
Handheld Programmer Keystrokes



**Or Positive Differential (ORPD)**

DS5	Implied
HPP	Used

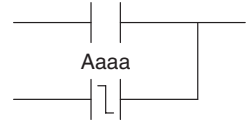
The Or Positive Differential instruction logically ors a contact in parallel with another contact in a rung. The status of the contact will be open until the associated image register point makes an Off-to-On transition, closing it for one CPU scan. Thereafter, it remains open until another Off-to-On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



**Or Negative Differential (ORND)**

DS5	Implied
HPP	Used

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.



5

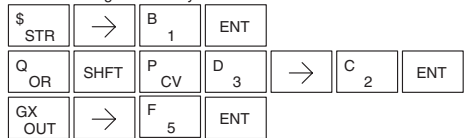
Operand Data Type	DL05 Range
..... A	aaa
Inputs ..... X	0-377
Outputs ..... Y	0-377
Control Relays ..... C	0-777
Stage ..... S	0-377
Timer ..... T	0-177
Counter ..... CT	0-177

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

DirectSOFT 5



Handheld Programmer Keystrokes

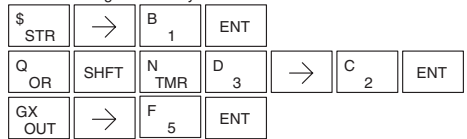


In the following example, Y 5 will energize whenever X1 is on, or for one CPU scan when X2 transitions from On to Off.

DirectSOFT 5



Handheld Programmer Keystrokes



### And Positive Differential (ANDPD)

DS5	Implied
HPP	Used

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



### And Negative Differential (ANDND)

DS5	Implied
HPP	Used

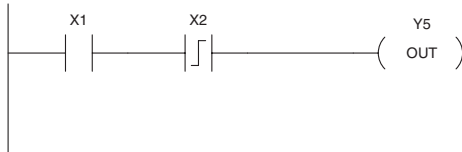
The And Negative Differential instruction logically ands a contact in series with another contact in a rung. The status of the contact will be open until the associated image register point makes an On-to-Off transition, closing it for one CPU scan. Thereafter, it remains open until another On-to-Off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



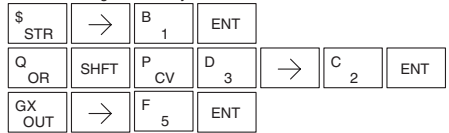
Operand Data Type	DL05 Range
..... A	aaa
Inputs ..... X	0-377
Outputs ..... Y	0-377
Control Relays ..... C	0-777
Stage ..... S	0-377
Timer ..... T	0-177
Counter ..... CT	0-177

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

DirectSOFT 5

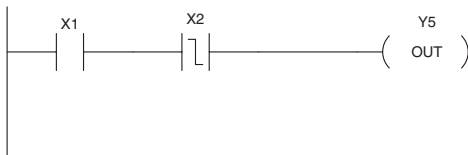


Handheld Programmer Keystrokes

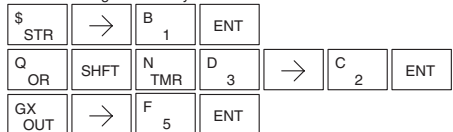


In the following example, Y5 will energize for one CPU scan whenever X1 is on and X2 transitions from On to Off.

DirectSOFT 5



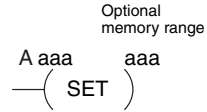
Handheld Programmer Keystrokes



**Set (SET)**

DS5	Used
HPP	Used

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)**

DS5	Used
HPP	Used

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.



5

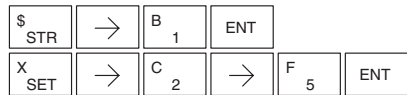
Operand Data Type	DL05 Range
..... A	aaa
Inputs ..... X	0-377
Outputs ..... Y	0-377
Control Relays ..... C	0-777
Stage ..... S	0-377
Timer ..... T	0-177
Counter ..... CT	0-177

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

DirectSOFT 5



Handheld Programmer Keystrokes

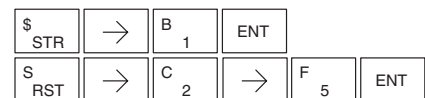


In the following example when X1 is on, Y2 through Y5 will be reset or de-energized.

DirectSOFT 5



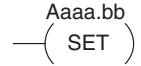
Handheld Programmer Keystrokes



### Set Bit-of-Word (SETB)

DS5	Used
HPP	Used

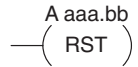
The Set Bit-of-Word instruction sets or turns on a bit in a V-memory location. Once the bit is set it will remain on until it is reset using the Reset Bit-of-Word instruction. It is not necessary for the input controlling the Set Bit-of-Word instruction to remain on.



### Reset Bit-of-Word (RSTB)

DS5	Used
HPP	Used

The Reset Bit-of-Word instruction resets or turns off a bit in a V-memory location. Once the bit is reset it is not necessary for the input to remain on.



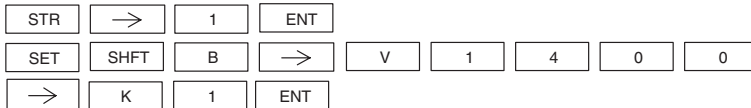
Operand Data Type		DL05 Range	
..... A		aaa	bb
V-memory .....	B	See memory map	BCD, 0 to 15
Pointer .....	PB	See memory map	BCD, 0 to 15

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

DirectSOFT 5



Handheld Programmer Keystrokes

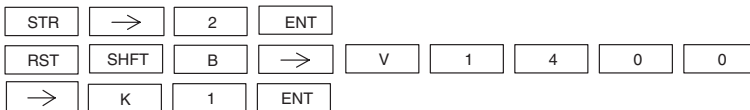


In the following example when X2 turns on, bit 1 in V1400 is reset to the off state.

DirectSOFT 5



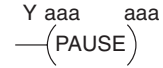
Handheld Programmer Keystrokes



**Pause (PAUSE)**

DS5	Used
HPP	Used

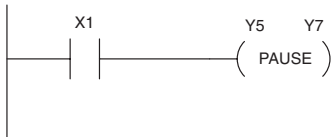
The Pause instruction disables the output update on a range of outputs. The ladder program will continue to run and update the image register. However, the outputs in the range specified in the Pause instruction will be turned off at the output points.



Operand Data Type	DL05 Range
	<b>aaa</b>
Outputs .....	0-377

In the following example, when X1 is ON, Y5-Y7 will be turned OFF. The execution of the ladder program will not be affected.

DirectSOFT 5



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.

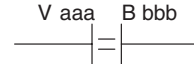


# Comparative Boolean

## Store If Equal (STRE)

DS5	Used
HPP	Used

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



## Store If Not Equal (STRNE)

DS5	Used
HPP	Used

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.



5

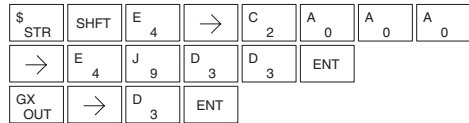
Operand Data Type	DL05 Range	
	aaa	bbb
..... B		
V-memory ..... V	All (See page 3-28)	All (See page 3-28)
Pointer ..... P	All (See page 3-28)	All (See page 3-28)
Constant ..... K	—	0-9999

In the following example, when the value in V-memory location V2000 = 4933, Y3 will energize.

DirectSOFT 5



Handheld Programmer Keystrokes

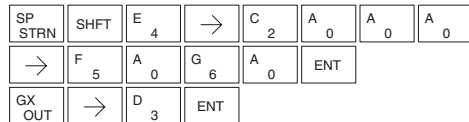


In the following example, when the value in V-memory location V2000 is not equal to 5060, Y3 will energize.

DirectSOFT 5



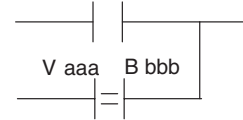
Handheld Programmer Keystrokes



**Or If Equal (ORE)**

DS5	Implied
HPP	Used

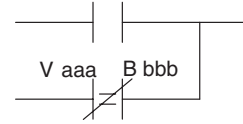
The Or If Equal instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when V<sub>aaa</sub> is equal to B<sub>bbb</sub>.



**Or If Not Equal (ORNE)**

DS5	Implied
HPP	Used

The Or If Not Equal instruction connects a normally closed comparative contact in parallel with another contact. The contact will be on when V<sub>aaa</sub> does not equal B<sub>bbb</sub>.

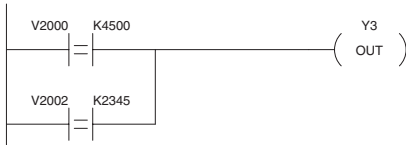


5

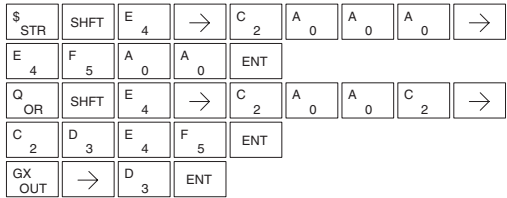
Operand Data Type	DL05 Range	
..... B	aaa	bbb
V-memory ..... V	All (See page 3-28)	All (See page 3-28)
Pointer ..... P	All (See page 3-28)	All (See page 3-28)
Constant ..... K	—	0-9999

In the following example, when the value in V-memory location V2000 = 4500 or V2002 = 2345, Y3 will energize.

DirectSOFT 5

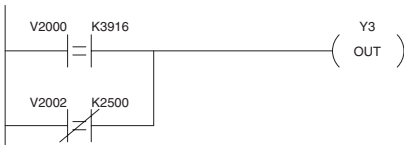


Handheld Programmer Keystrokes

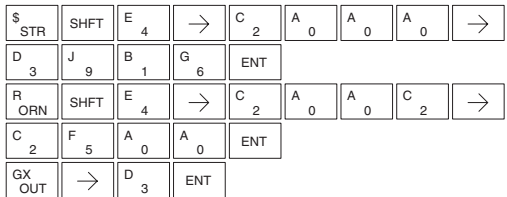


In the following example, when the value in V-memory location V2000 = 3916 or V2002 is not equal to 2500, Y3 will energize.

DirectSOFT 5



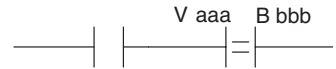
Handheld Programmer Keystrokes



### And If Equal (ANDE)

DS5	Implied
HPP	Used

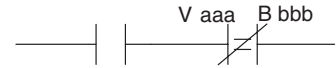
The And If Equal instruction connects a normally open comparative contact in series with another contact. The contact will be on when Vaaa is equal to Bbbb.



### And If Not Equal (ANDNE)

DS5	Implied
HPP	Used

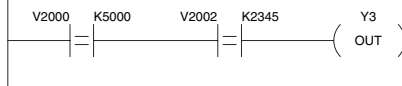
The And If Not Equal instruction connects a normally closed comparative contact in series with another contact. The contact will be on when Vaaa does not equal Bbbb.



Operand Data Type	DL05 Range	
	aaa	bbb
..... A/B		
V-memory ..... V	All (See page 3-28)	All (See page 3-28)
Pointer ..... P	All (See page 3-28)	All (See page 3-28)
Constant ..... K	—	0-9999

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

DirectSOFT 5

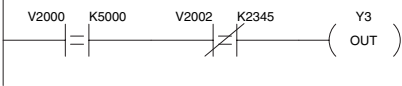


Handheld Programmer Keystrokes

\$	STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
F 5	A 0	A 0	A 0	ENT					
V	AND	SHFT	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 = 2550 and V2002 does not equal 2345, Y3 will energize.

DirectSOFT 5



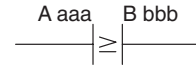
Handheld Programmer Keystrokes

\$	STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
F 5	A 0	A 0	A 0	ENT					
V	AND	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT					
GX	OUT	→	D 3	ENT					

**Store (STR)**

DS5	Used
HPP	Used

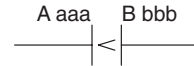
The Comparative Store instruction begins a new rung or additional branch in a rung with a normally open comparative contact. The contact will be on when Aaaa is equal to or greater than Bbbb.



**Store Not (STRN)**

DS5	Used
HPP	Used

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

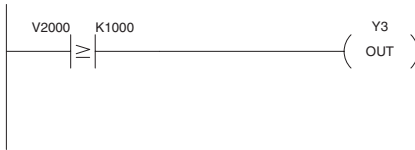


5

Operand Data Type	DL05 Range	
	aaa	bbb
..... A/B		
V-memory ..... V	All (See page 3-28)	All (See page 3-28)
Pointer ..... P	All (See page 3-28)	All (See page 3-28)
Constant ..... K	—	0-9999
Timer ..... T	0-177	
Counter ..... CT	0-177	

In the following example, when the value in V-memory location V2000 ≥ 1000, Y3 will energize.

DirectSOFT 5

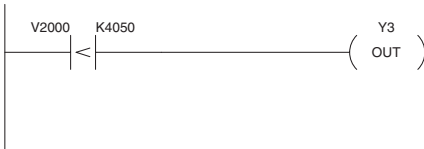


Handheld Programmer Keystrokes

\$ 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.

DirectSOFT 5



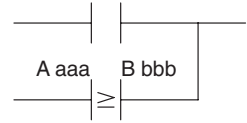
Handheld Programmer Keystrokes

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)

DS5	Implied
HPP	Used

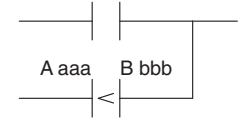
The Comparative Or instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa is equal to or greater than Bbbb.



### Or Not (ORN)

DS5	Implied
HPP	Used

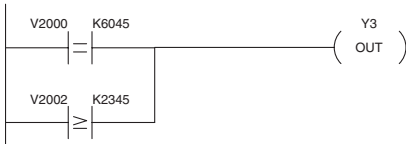
The Comparative Or Not instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa is less than Bbbb.



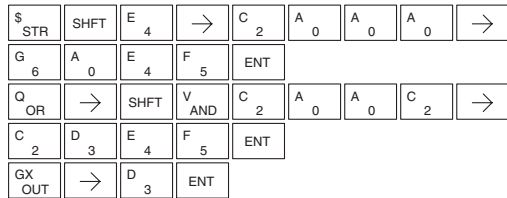
Operand Data Type	DL05 Range	
	aaa	bbb
..... A/B		
V-memory .....	All (See page 3-28)	All (See page 3-28)
Pointer .....	All (See page 3-28)	All (See page 3-28)
Constant .....	—	0-9999
Timer .....	0-177	
Counter .....	0-177	

In the following example, when the value in V-memory location V2000 = 6045 or V2002 ≥ 2345, Y3 will energize.

DirectSOFT 5

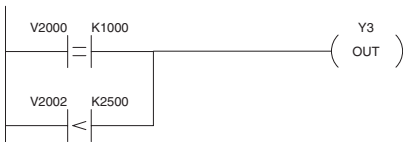


Handheld Programmer Keystrokes

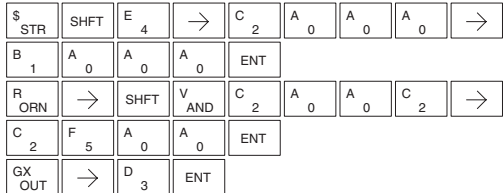


In the following example when the value in V-memory location V2000 = 1000 or V2002 < 2500, Y3 will energize.

DirectSOFT 5



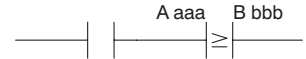
Handheld Programmer Keystrokes



**And (AND)**

DS5	Implied
HPP	Used

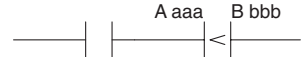
The Comparative And instruction connects a normally open comparative contact in series with another contact. The contact will be on when Aaaa is equal to or greater than Bbbb.



**And Not (ANDN)**

DS5	Implied
HPP	Used

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

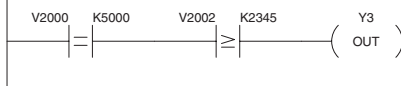


5

Operand Data Type	DL05 Range	
	aaa	bbb
..... A/B		
V-memory ..... V	All (See page 3-28)	All (See page 3-28)
Pointer ..... p	All (See page 3-28)	All (See page 3-28)
Constant ..... K	—	0-9999
Timer ..... T	0-177	
Counter ..... CT	0-177	

In the following example, when the value in V-memory location V2000 = 5000, and V2002 ≥ 2345, Y3 will energize.

DirectSOFT 5

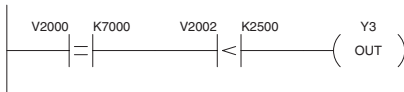


Handheld Programmer Keystrokes

\$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
F 5	A 0	A 0	A 0	ENT				
V AND	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

In the following example, when the value in V-memory location V2000 = 7000 and V2002 < 050, Y3 will energize.

DirectSOFT 5



Handheld Programmer Keystrokes

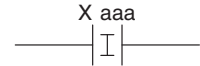
\$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
H 7	A 0	A 0	A 0	ENT				
W ANDN	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	D 3	ENT					

# Immediate Instructions

## Store Immediate (STRI)

DS5	Used
HPP	Used

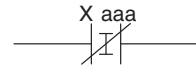
The Store Immediate instruction begins a new rung or additional branch in a rung. The status of the contact will be the same as the status of the associated input point *at the time the instruction is executed*. The image register is not updated.



## Store Not Immediate (STRNI)

DS5	Used
HPP	Used

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.



5

Operand Data Type	DL05 Range
	<b>aaa</b>
Inputs .....X	0-377

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

DirectSOFT 5



Handheld Programmer Keystrokes

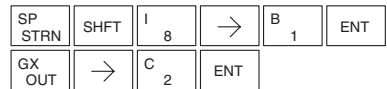


In the following example when X1 is off, Y2 will energize.

DirectSOFT 5



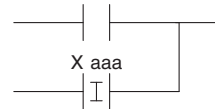
Handheld Programmer Keystrokes



## Or Immediate (ORI)

DS5	Implied
HPP	Used

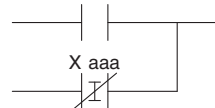
The Or Immediate connects two contacts in parallel. The status of the contact will be the same as the status of the associated input point *at the time the instruction is executed*. The image register is not updated.



## Or Not Immediate (ORNI)

DS5	Implied
HPP	Used

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.

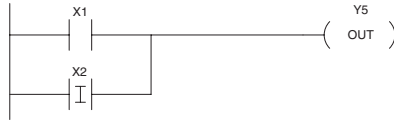


OR Immediate Instructions (cont'd)

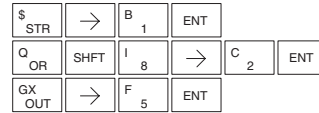
Operand Data Type	DL05 Range
	<b>aaa</b>
Inputs .....X	0-377

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

DirectSOFT 5

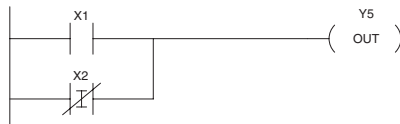


Handheld Programmer Keystrokes

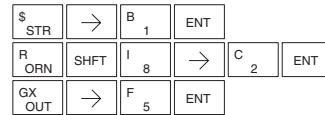


In the following example, when X1 is on or X2 is off, Y5 will energize.

DirectSOFT 5



Handheld Programmer Keystrokes



And Immediate (ANDI)

DS5	Implied
HPP	Used

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)

DS5	Implied
HPP	Used

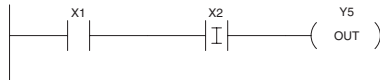
The And Not Immediate connects two contacts in series. The status of the contact will be opposite the status of the associated input point *at the time the instruction is executed*. The image register is not updated.



Operand Data Type	DL05 Range
	<b>aaa</b>
Inputs .....X	0-377

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

DirectSOFT 5

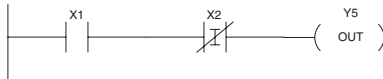


Handheld Programmer Keystrokes

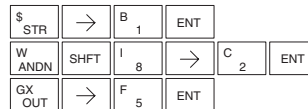


In the following example, when X1 is on and X2 is off, Y5 will energize.

DirectSOFT 5



Handheld Programmer Keystrokes

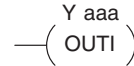




### Out Immediate (OUTI)

DS5	Used
HPP	Used

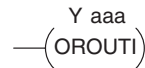
The Out Immediate instruction reflects the status of the rung (on/off) and outputs the discrete (on/off) status to the specified module output point and the image register *at the time the instruction is executed*. If multiple Out Immediate instructions referencing the same discrete point are used it is possible for the module output status to change multiple times in a CPU scan. See Or Out Immediate.



### Or Out Immediate (OROUTI)

DS5	Used
HPP	Used

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.



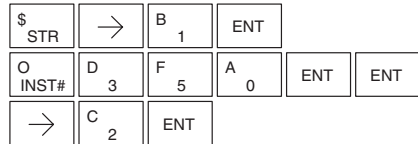
Operand Data Type	DL05 Range
	<b>aaa</b>
Outputs .....Y	0-377

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.

DirectSOFT 5

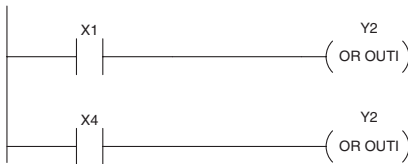


Handheld Programmer Keystrokes

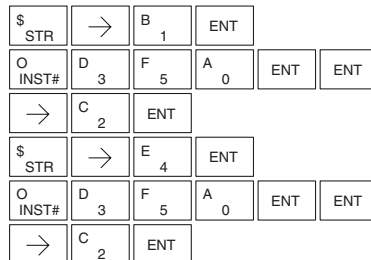


In the following example, when X1 or X4 is on, Y2 will energize.

DirectSOFT 5



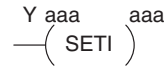
Handheld Programmer Keystrokes



### Set Immediate (SETI)

DS5	Used
HPP	Used

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)

5

DS5	Used
HPP	Used

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.



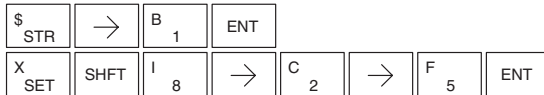
Operand Data Type	DL05 Range
	<b>aaa</b>
Outputs .....Y	0-377

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.

DirectSOFT 5

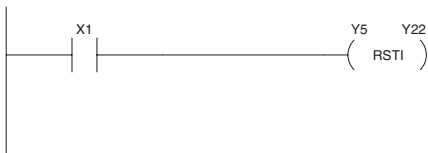


Handheld Programmer Keystrokes

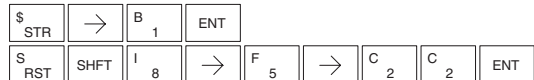


In the following example, when X1 is on, Y5 through Y22 will be reset (off) in the image register and on the corresponding output module(s).

DirectSOFT 5



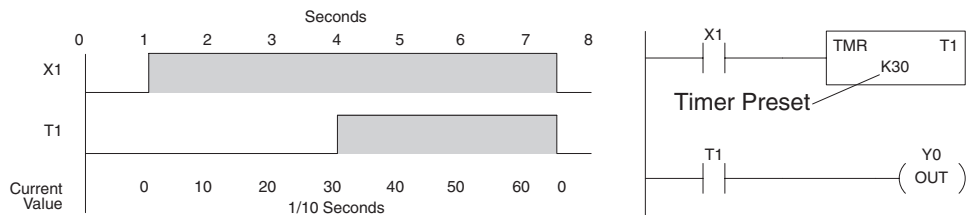
Handheld Programmer Keystrokes



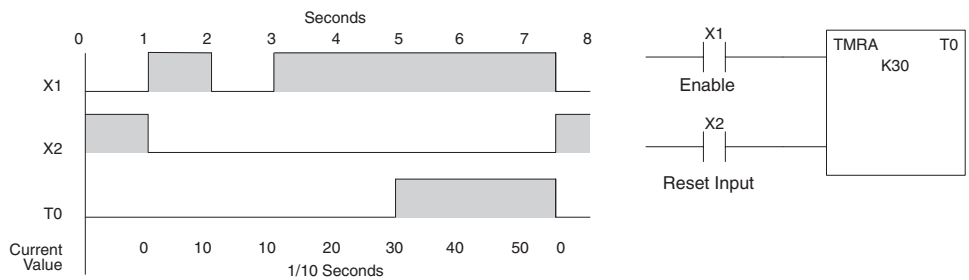
## Timer, Counter and Shift Register Instructions

### 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.



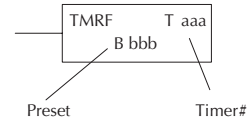
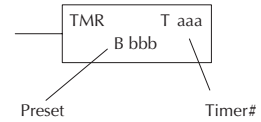
There are those applications that need an accumulating timer, meaning it has the ability to time, stop, and then resume from where it previously stopped. The accumulating timer works similarly to the regular timer, but two inputs are required. The start/stop input starts and stops the timer. When the timer stops, the elapsed time is maintained. When the timer starts again, the timing continues from the elapsed time. When the reset input is turned on, the elapsed time is cleared and the timer will start at 0 when it is restarted. There is a tenth of a second and a hundredth of a second timer available with a maximum time of 9999999.9 and 999999.99 seconds respectively. The timing diagram below shows the relationship between the timer input, timer reset, associated discrete bit, current value, and timer preset.



### Timer (TMR) and Timer Fast (TMRF)

DS5	Used
HPP	Used

The Timer instruction is a 0.1 second single input timer that times to a maximum of 999.9 seconds. The Timer Fast instruction is a 0.01 second single input timer that times up to a maximum of 99.99 seconds. These timers will be enabled if the input logic is true (on) and will be reset to 0 if the input logic is false (off).



The timer discrete status bit and the current value are not specified in the timer instruction

#### Instruction Specifications

**Timer Reference (Taaa):** Specifies the timer number.

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

**Current Value:** Timer current values (BCD) 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 as a BCD value.

**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 TA2.

5



**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	DL05 Range	
	aaa	bbb
Timers ..... A/B	aaa	bbb
Timers ..... T	0-177	—
V-memory for preset values ..... V	—	1200-7377 7400-7577*
Pointers (preset only) ..... P	—	1200-7377 7400-7577
Constants (preset only) ..... K	—	0-9999
Timer discrete status bits ..... T/V	0-177 or V41100-41107	
Timer current values ..... V/T**	0-177	



**NOTE:** \* May be non-volatile if MOV instruction is used.

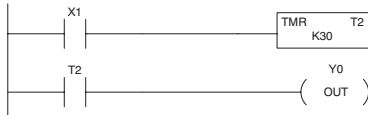
\*\* With the HPP, both the Timer discrete status bits and current value are accessed with the same data reference. **DirectSOFT 5** uses separate references, such as “T2” for discrete status bit for Timer T2, and “TA2” for the current value of Timer T2.

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

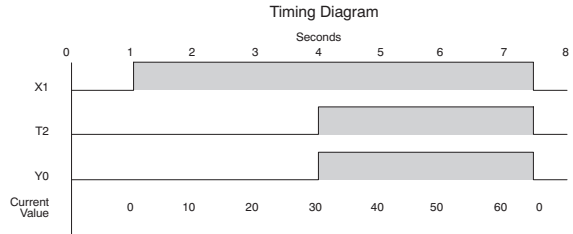
In the following example, a single input timer is used with a preset of 3 seconds. The timer discrete status bit (T2) will turn on when the timer has timed for 3 seconds. The timer is reset when X1 turns off, turns off the discrete status bit and resets the timer current value to 0.

DirectSOFT 5



Handheld Programmer Keystrokes

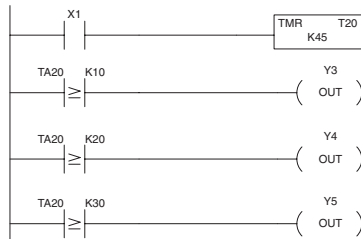
\$ STR	→	B 1	ENT
N TMR	→	C 2	→ D 3 A 0 ENT
\$ STR	→	SHFT T MLR	C 2 ENT
GX OUT	→	A 0	ENT



### Timer Example Using Comparative Contacts

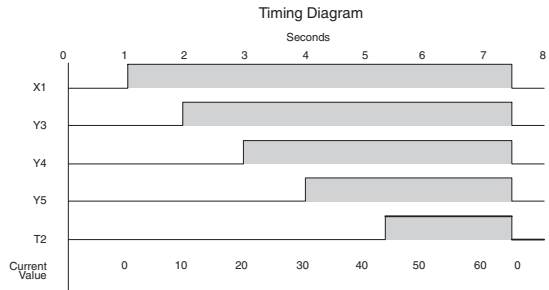
In the following example, a single input timer is used with a preset of 4.5 seconds. Comparative contacts are used to energize Y3, Y4, and Y5 at one second intervals respectively. When X1 is turned off the timer will be reset to 0 and the comparative contacts will turn off Y3, Y4, and Y5.

DirectSOFT 5



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
N TMR	→	C 2	→ A 0 → E 4 F 5 ENT
\$ STR	→	SHFT T MLR	C 2 A 0 → B 1 A 0 ENT
GX OUT	→	D 3	ENT
\$ STR	→	SHFT T MLR	C 2 A 0 → C 2 A 0 ENT
GX OUT	→	E 4	ENT
\$ STR	→	SHFT T MLR	C 2 A 0 → D 3 A 0 ENT
GX OUT	→	F 5	ENT



### Accumulating Timer (TMRA)

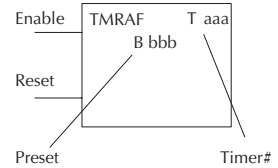
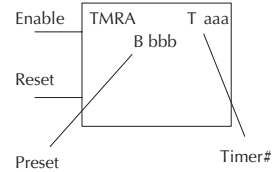
DS5	Used
HPP	Used

The Accumulating Timer is a 0.1 second two input timer that will time to a maximum of 9999999.9.

### Accumulating Fast Timer (TMRAF)

DS5	Used
HPP	Used

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.



**The timer discrete status bit and the current value are not specified in the timer instruction**

#### Instruction Specifications

**Timer Reference (Taaa):** Specifies the timer number.

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

**Current Value:** Timer current values (BCD) 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 as a BCD value.

**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 TMR 1 is used, the next available timer number is TMR 3.

Operand Data Type	DL05 Range	
	aaa	bbb
..... A/B	aaa	bbb
Timers ..... T	0-176	—
V-memory for preset values ..... V	—	1200-7377/7400-7577*
Pointers (preset only) ..... P	—	1200-7377/7400-7577
Constants (preset only) ..... K	—	0-99999999
Timer discrete status bits ..... T/V	0-176 or V41100-41107	
Timer current values ..... V/T**	0-176	



**NOTE:** \* May be non-volatile if MOV instruction is used.

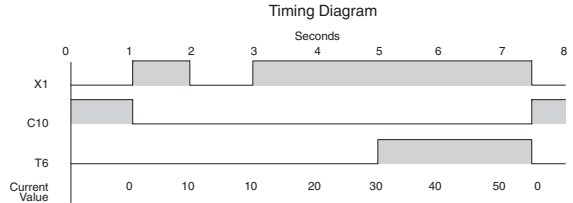
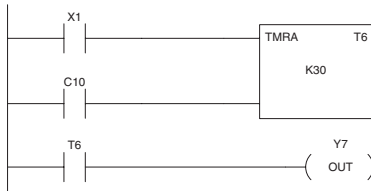
\*\* With the HPP, both the Timer discrete status bits and current value are accessed with the same data reference. **DirectSOFT 5** uses separate references, such as “T2” for discrete status bit for Timer T2, and “TA2” for the current value of Timer T2.

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.

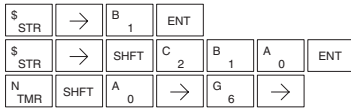
### Accumulating Timer Example using Discrete Status Bits

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.

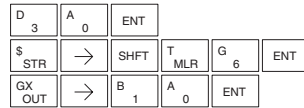
DirectSOFT 5



Handheld Programmer Keystrokes



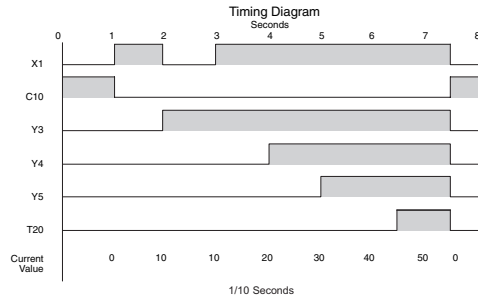
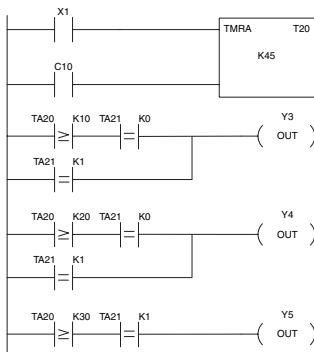
Handheld Programmer Keystrokes (cont)



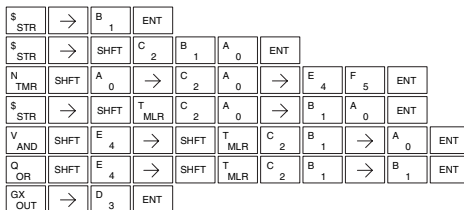
### Accumulator Timer Example Using Comparative Contacts

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.

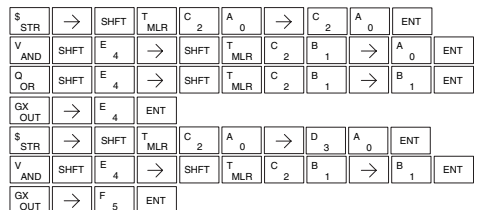
Contacts  
DirectSOFT



Handheld Programmer Keystrokes



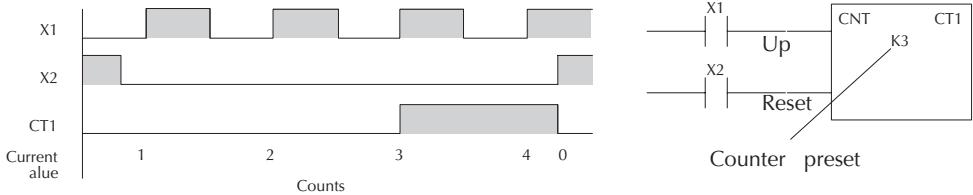
Handheld Programmer Keystrokes (cont'd)



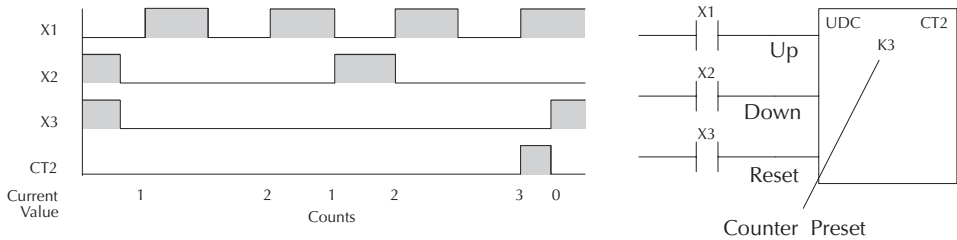
### Using Counters

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).

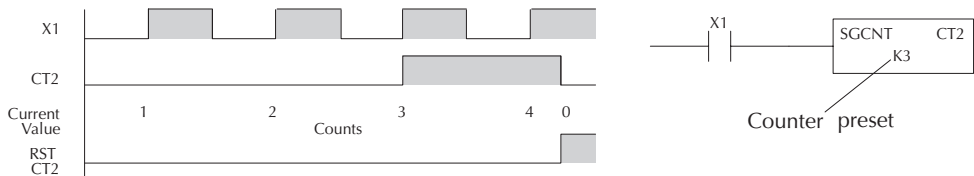
The up counter has two inputs, a count input and a reset input. The maximum count value is 9999. The timing diagram below shows the relationship between the counter input, counter reset, associated discrete bit, current value, and counter preset.



The up down counter has three inputs, a count up input, count down input and reset input. The maximum count value is 99999999. The timing diagram below shows the relationship between the counter input, counter reset, associated discrete bit, current value, and counter preset.



The stage counter has a count input and is reset by the RST instruction. This instruction is useful when programming 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.

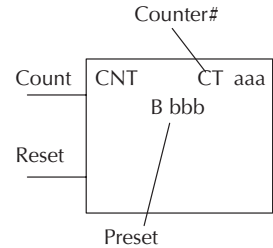




### Counter (CNT)

DS5	Used
HPP	Used

The Counter is a two input counter that increments when the count input logic transitions from off to on. When the counter reset input is on the counter resets to 0. When the current value equals the preset value, the counter status bit comes on and the counter continues to count up to a maximum count of 9999. The maximum value will be held until the counter is reset.



The counter discrete status bit and the current value are not specified in the counter instruction.

5

#### Instruction Specifications

**Counter Reference (CTaaa):** Specifies the counter number.

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

**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 as a BCD value.

**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	DL05 Range	
	aaa	bbb
..... A/B	aaa	bbb
Counters ..... CT	0-177	—
V-memory (preset only) ..... V	—	1200-7377 7400-7577*
Pointers (preset only) ..... P	—	1200-7377 7400-7577
Constants (preset only) ..... K	—	0-9999
Counter discrete status bits ..... CT/V	0-177 or V41140-41147	
Counter current values ..... V /CT**	0-177	



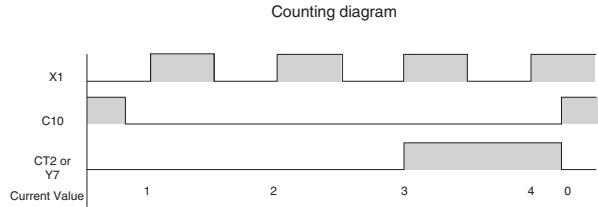
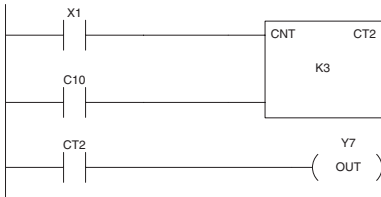
**NOTE:** \* May be non-volatile if MOV instruction is used.

\*\* With the HPP, both the Counter discrete status bits and current value are accessed with the same data reference. **DirectSOFT 5** 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.

DirectSOFT 5



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
\$ STR	→	SHFT C 2	B 1 A 0 ENT
GY CNT	→	C 2	→ D 3 ENT

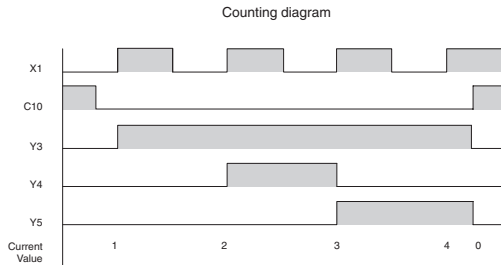
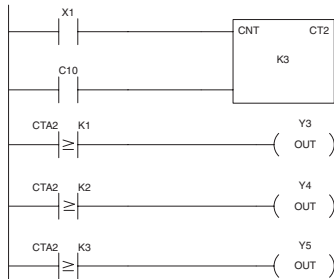
Handheld Programmer Keystrokes (cont)

\$ STR	→	SHFT C 2	SHFT T MLR	C 2	ENT
GX OUT	→	B 1	A 0	ENT	

### Counter Example Using Comparative Contacts

In the following example, when X1 makes an off to on transition, counter CT2 will increment by one. Comparative contacts are used to energize Y3, Y4, and Y5 at different counts. When the reset C10 turns on, the counter status bit will turn off and the counter current value will be 0, and the comparative contacts will turn off.

DirectSOFT 5



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
\$ STR	→	SHFT C 2	B 1 A 0 ENT
GY CNT	→	C 2	→ D 3 ENT
\$ STR	→	SHFT C 2	SHFT T MLR C 2
→	B 1	ENT	
GX OUT	→	D 3	ENT

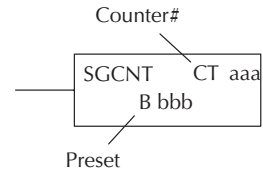
Handheld Programmer Keystrokes (cont)

\$ STR	→	SHFT C 2	SHFT T MLR	C 2
→	C 2	ENT		
GX OUT	→	E 4	ENT	
\$ STR	→	SHFT C 2	SHFT T MLR	C 2
→	D 3	ENT		
GX OUT	→	F 5	ENT	

### Stage Counter (SGCNT)

DS5	Used
HPP	Used

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.



The counter discrete status bit and the current value are not specified in the counter instruction.

5

#### 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	DL05 Range	
	aaa	bbb
..... A/B		
Counters ..... CT	0-177	—
V-memory (preset only) ..... V	—	1200-7377 7400-7577*
Pointers (preset only) ..... P	—	1200-7377 7400-7577
Constants (preset only) ..... K	—	0-9999
Counter discrete status bits ..... CT/N	0-177 or V41140-41147	
Counter current values ..... V/CT**	1000-1177	



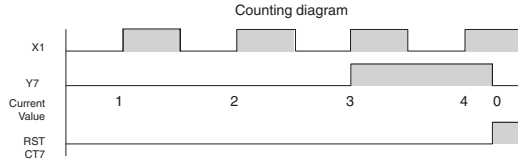
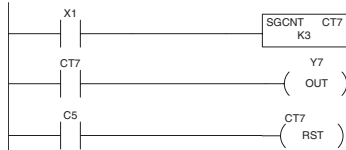
**NOTE:** \* May be non-volatile if MOV instruction is used.

\*\* With the HPP, both the Counter discrete status bits and current value are accessed with the same data reference. **DirectSOFT 5** uses separate references, such as "CT2" for discrete status bit for Counter CT2, and "CTA2" for the current value of Counter CT2.

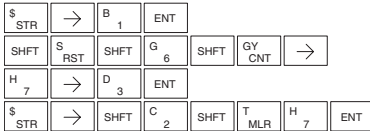
### Stage Counter Example Using Discrete Status Bits

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.

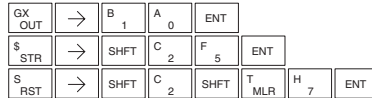
DirectSOFT 5



Handheld Programmer Keystrokes



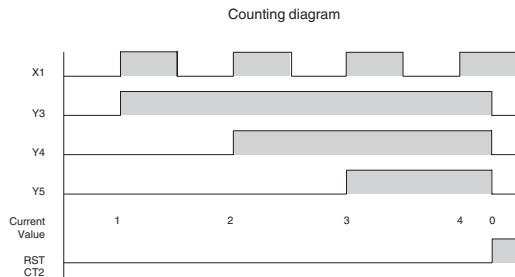
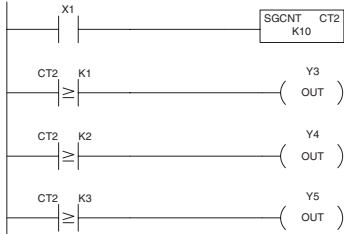
Handheld Programmer Keystrokes (cont)



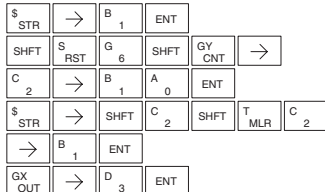
### Stage Counter Example Using Comparative Contacts

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.

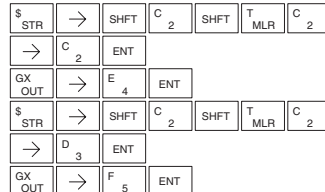
DirectSOFT 5



Handheld Programmer Keystrokes



Handheld Programmer Keystrokes (cont)



### Up Down Counter (UDC)

DS5	Used
HPP	Used

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.

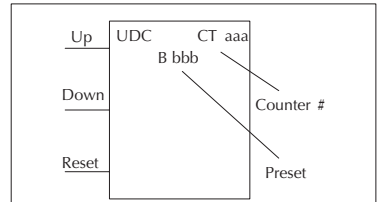
#### Instruction Specification

**Counter Reference (CTaaa):** Specifies the counter number.

**Preset Value (Bbbb):** Constant value (K) or two consecutive V-memory locations as a BCD value.

**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 as a BCD value.

**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.



Caution: The UDC uses two V-memory locations for the 8 digit current value. This means that the UDC uses two consecutive counter locations. If UDC CT1 is used in the program, the next available counter is CT3.

The counter discrete status bit and the current value are not specified in the counter instruction

Operand Data Type	DL05 Range	
A/B	aaa	bbb
Counters ..... CT	0–176	—
V-memory (preset only) ..... V	—	1200–7377 7400–7577*
Pointers (preset only) ..... P	—	1200–7377 7400–7577
Constants (preset only) ..... K	—	0–99999999
Counter discrete status bits ..... CT/V	0–176 or V41140–41147	
Counter current values ..... V/CT**	0–176	



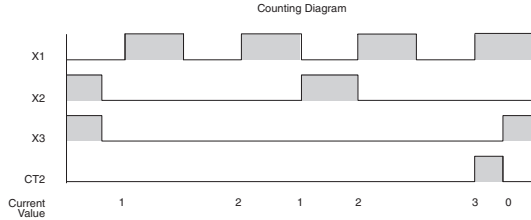
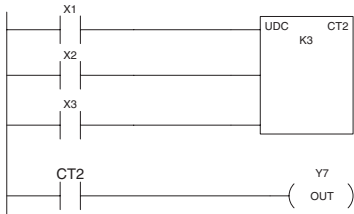
**NOTE:** \* May be non-volatile if MOV instruction is used.

\*\* With the HPP, both the Counter discrete status bits and current value are accessed with the same data reference. **DirectSOFT 5** 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.

DirectSOFT 5



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT		
\$ STR	→	C 2	ENT		
\$ STR	→	D 3	ENT		
SHFT	U ISG	D 3	C 2	→	C 2

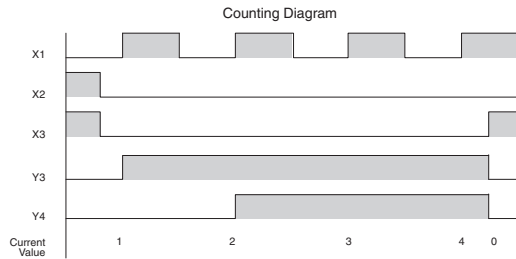
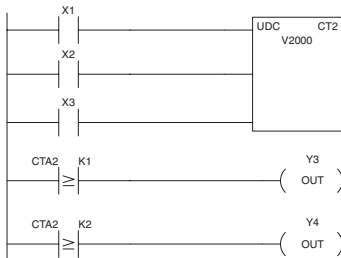
Handheld Programmer Keystrokes (cont)

→	D 3	ENT					
\$ STR	→	SHFT	C 2	SHFT	T MLR	C 2	ENT
GX OUT	→	B 1	A 0	ENT			

### Up / Down Counter Example Using Comparative Contacts

In the following example, when X1 makes an off to on transition, counter CT2 will increment by one. Comparative contacts are used to energize Y3 and Y4 at different counts. When the reset (X3) turns on, the counter status bit will turn off, the current value will be 0, and the comparative contacts will turn off.

DirectSOFT 5



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT			
\$ STR	→	C 2	ENT			
\$ STR	→	D 3	ENT			
SHFT	U ISG	D 3	C 2	→	C 2	→
SHFT	V AND	C 2	A 0	A 0	A 0	ENT
\$ STR	→	SHFT	C 2	SHFT	T MLR	C 2

Handheld Programmer Keystrokes (cont)

→	B 1	ENT				
GX OUT	→	D 3	ENT			
\$ STR	→	SHFT	C 2	SHFT	T MLR	C 2
→	C 2	ENT				
GX OUT	→	E 4	ENT			

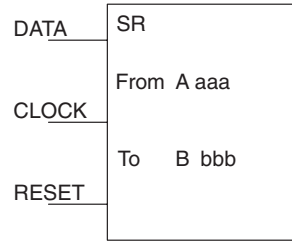
### Shift Register (SR)

DS5	Used
HPP	Used

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

The Shift Register has three contacts.

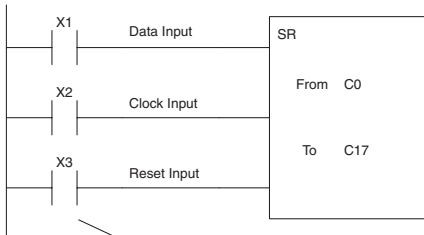
- Data — determines the value (1 or 0) that will enter the register
- Clock — shifts the bits one position on each low to high transition
- Reset —resets the Shift Register to all zeros.



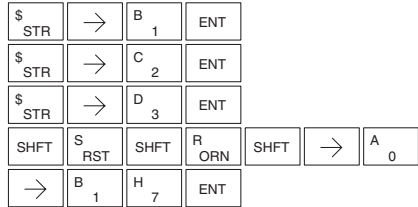
With each off to on transition of the clock input, the bits which make up the shift register block are shifted by one bit position and the status of the data input is placed into the starting bit position in the shift register. The direction of the shift depends on the entry in the From and To fields. 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	DL05 Range	
..... A/B	aaa	bbb
Control Relay ..... C	0-777	0-777

DirectSOFT 5

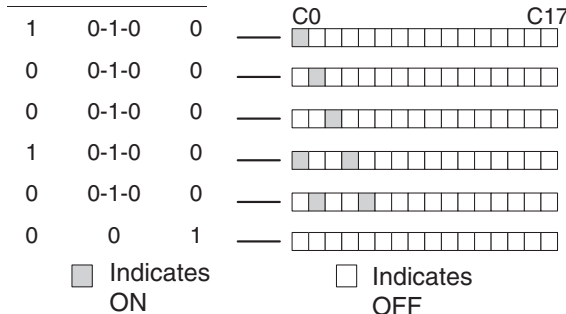


Handheld Programmer Keystrokes



Inputs on Successive Scans

Shift Register Bits



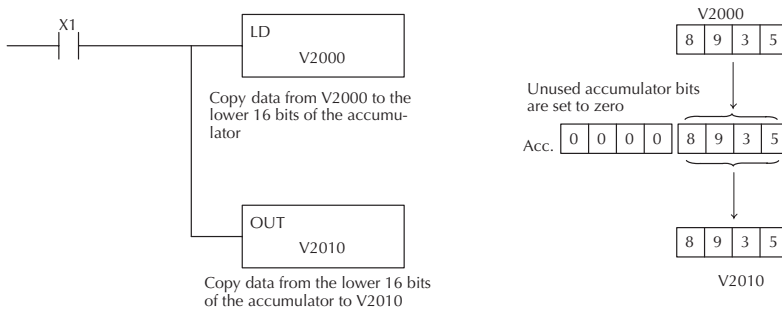
# Accumulator/Stack Load and Output Data Instructions

## Using the Accumulator

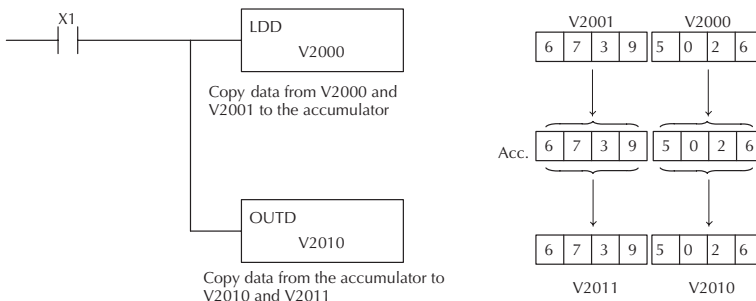
The accumulator in the DL05 internal CPUs is a 32 bit register which is used as a temporary storage location for data that is being copied or manipulated in some manor. For example, you have to use the accumulator to perform math operations such as add, subtract, multiply, etc. Since there are 32 bits, 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.



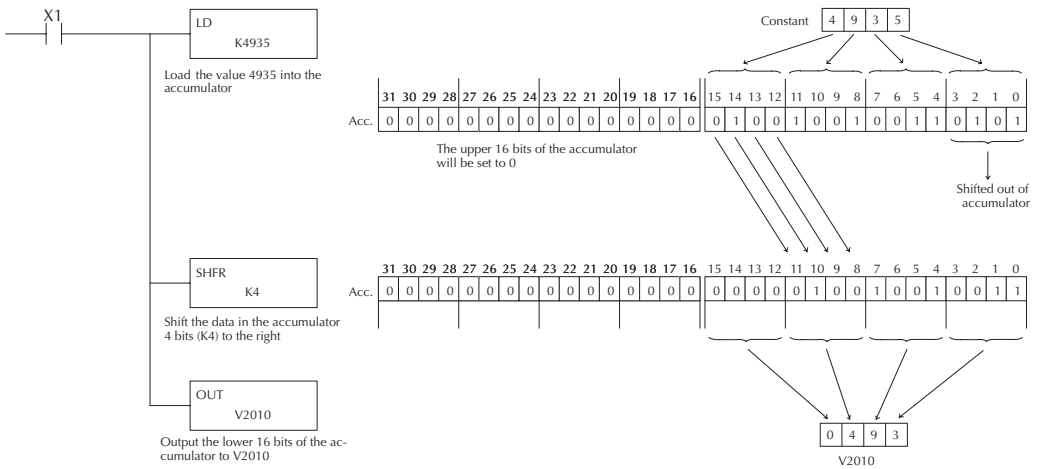
Since the accumulator is 32 bits and V-memory locations are 16 bits the Load Double and Out Double (or variations thereof) use two consecutive V-memory locations or 8 digit BCD constants to copy data either to the accumulator from a V-memory address or from a V-memory address to the accumulator. For 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

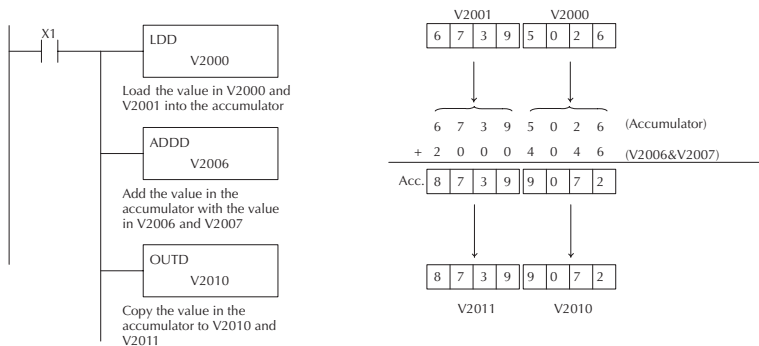
Instructions that manipulate data also use the accumulator. The result of the manipulated data resides in the accumulator. The data that was being manipulated is cleared from the accumulator. The following example loads the constant value 4935 into the accumulator, shifts the data right 4 bits, and outputs the result to V2010.



5

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.

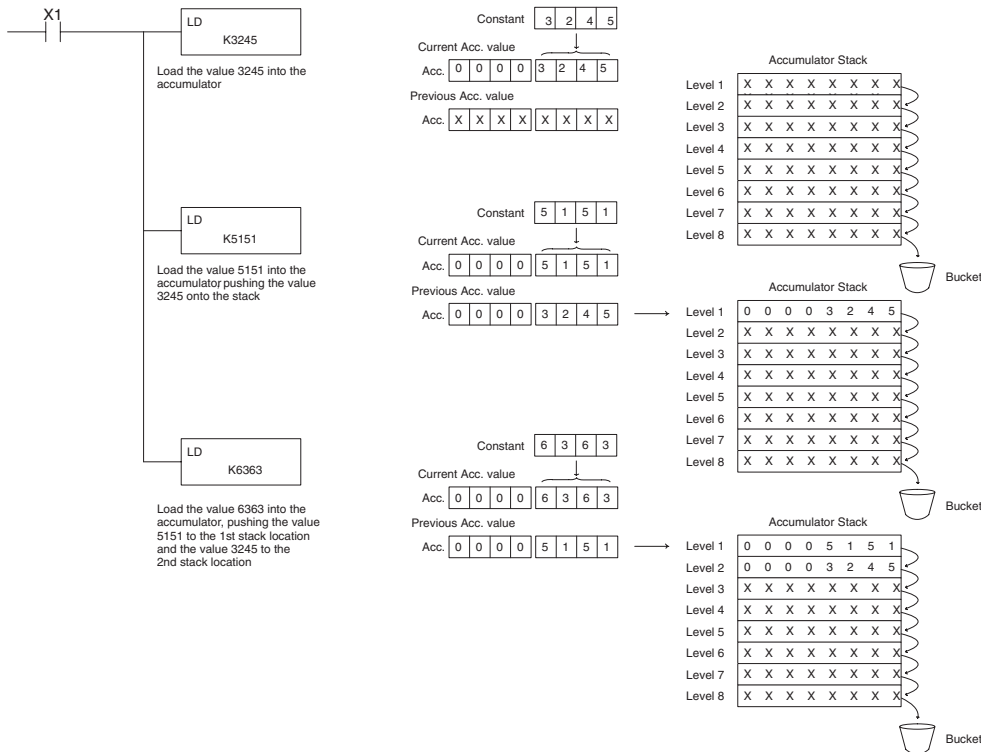
In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is added with the value in V2006 and V2007 using the Add Double instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.



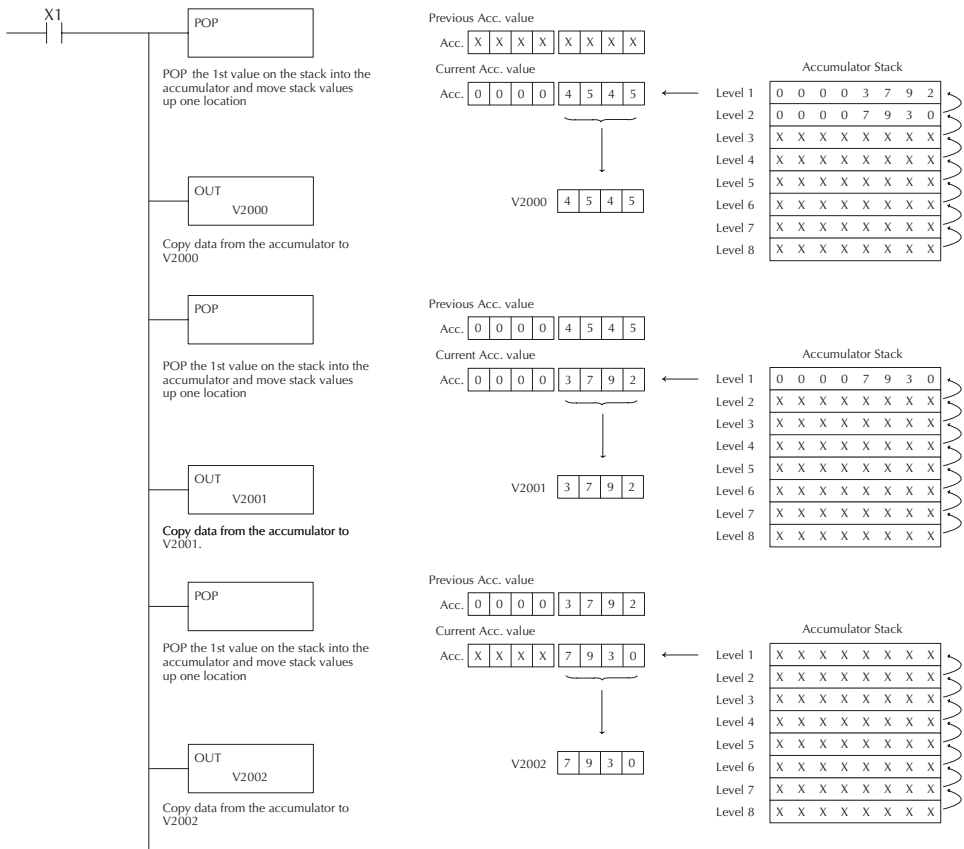
## Using the Accumulator Stack

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.

5



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.



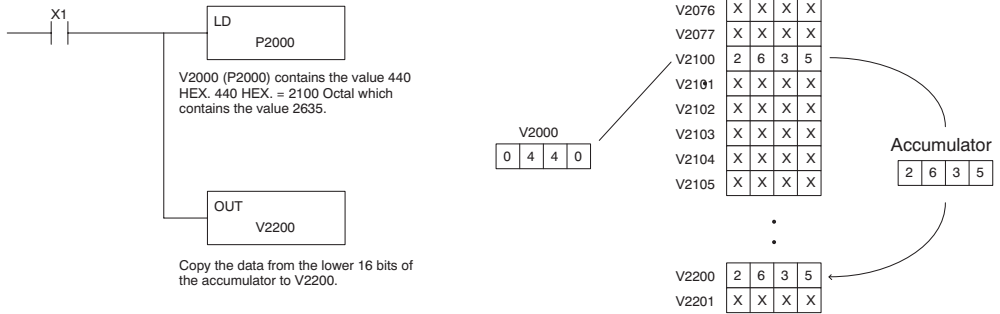
## Using Pointers

Many of the DL05 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.

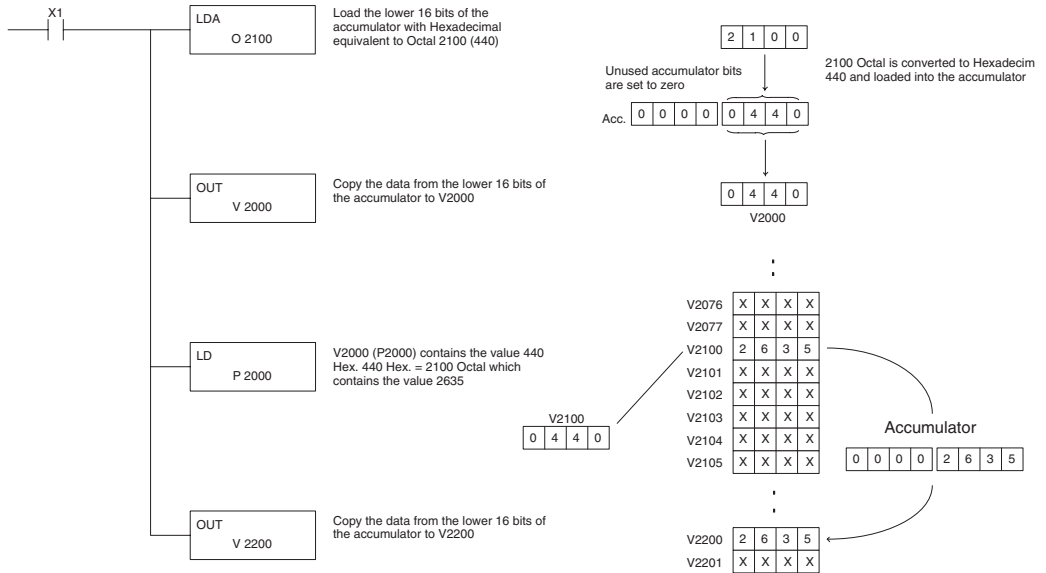


**NOTE:** DL05 V-memory addressing is in octal. However, the pointers reference a V-memory location with values viewed as HEX. Use the Load Address (LDA) instruction to move an address into the pointer location. This instruction performs the Octal to Hexadecimal conversion automatically.

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.



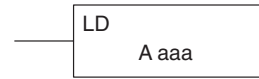
The following example is identical to the one above with one exception. The LDA (Load Address) instruction automatically converts the Octal address to Hex.



### Load (LD)

DS5	Used
HPP	Used

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	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0–FFFF

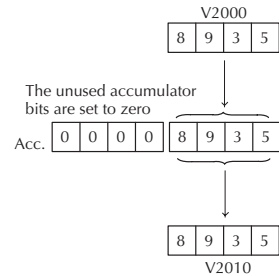
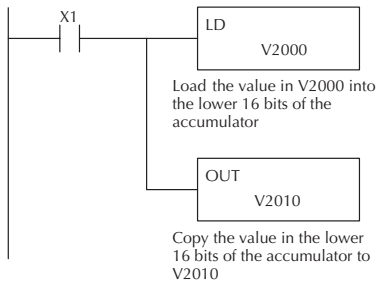
Discrete Bit Flags	Description
SP53	On when the pointer is outside of the available range.
SP70	On anytime the value in the accumulator is negative.
SP76	On when the value loaded into the accumulator is zero.



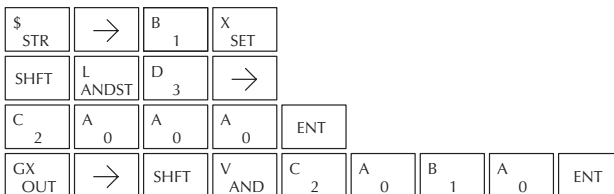
**NOTE:** Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.

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

DirectSOFT 5



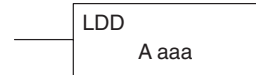
Handheld Programmer Keystrokes



### Load Double (LDD)

DS5	Used
HPP	Used

The Load Double instruction is a 32 bit instruction that loads the value (Aaaa), which is either two consecutive V-memory locations or an 8 digit constant value, into the accumulator.



Operand Data Type	DL05 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 the value loaded into the accumulator by any instruction is zero.

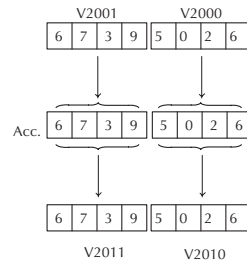
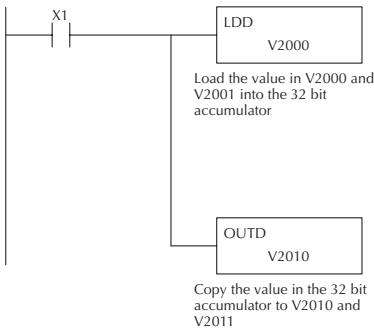
5



**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.

DirectSOFT 5



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT	
SHIFT	L ANDST	D 3	D 3	→
C 2	A 0	A 0	A 0	ENT
GX OUT	SHIFT	D 3	→	
C 2	A 0	B 1	A 0	ENT

### Load Formatted (LDF)

DS5	Used
HPP	Used

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		DL05 Range	
A		aaa	bbb
Inputs	X	0–377	—
Outputs	Y	0–377	—
Control Relays	C	0–777	—
Stage Bits	S	0–377	—
Timer Bits	T	0–177	—
Counter Bits	CT	0–177	—
Special Relays	SP	0–777	—
Constant	K	—	1–32

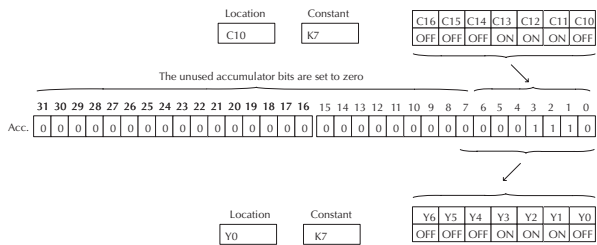
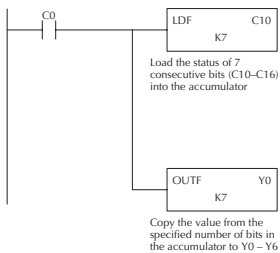
Discrete Bit Flags	Description
SP70	On anytime the value in the accumulator is negative.
SP76	On when the value loaded into the accumulator by any instruction is zero.



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

In the following example, when C0 is on, the binary pattern of C10–C16 (7 bits) will be loaded into the accumulator using the Load Formatted instruction. The lower 7 bits of the accumulator are output to Y0–Y6 using the Out Formatted instruction.

**DirectSOFT 5**



**Handheld Programmer Keystrokes**

\$ STR	→	SHFT	C 2	A 0	ENT
SHFT	L ANDST	D 3	F 5	→	
SHFT	C 2	B 1	A 0	→	H 7 ENT
GX OUT	SHFT	F 5	→		
A 0	→	H 7	ENT		

### Load Address (LDA)

DS5	Used
HPP	Used

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 DL05 system are in octal.



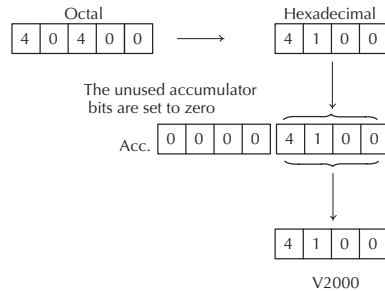
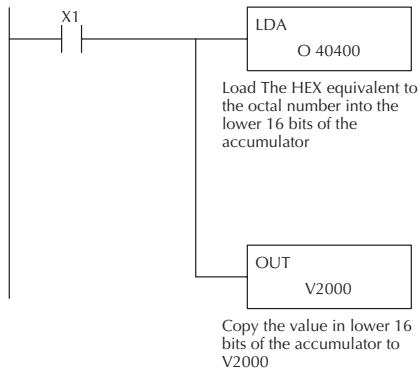
Operand Data Type	DL05 Range
	<b>aaa</b>
Octal Address ..... 0	See memory map

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

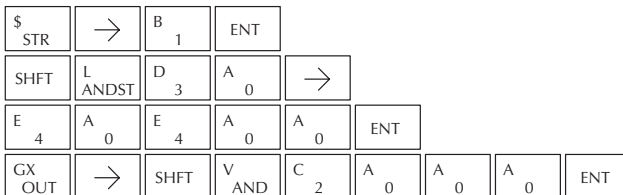
**NOTE:** Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.

In the following example when X1 is on, the octal number 40400 will be converted to a HEX 4100 and loaded into the accumulator using the Load Address instruction. The value in the lower 16 bits of the accumulator is copied to V2000 using the Out instruction.

DirectSOFT 5



Handheld Programmer Keystrokes

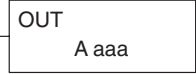




### Out (OUT)

DS5	Used
HPP	Used

The Out instruction is a 16 bit instruction that copies the value in the lower 16 bits of the accumulator to a specified V-memory location (Aaaa).



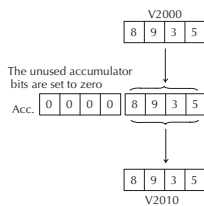
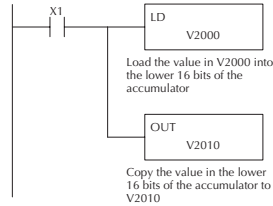
Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map

Discrete Bit Flags	Description
SP53	On when the pointer is outside of the available range.

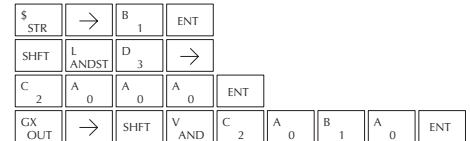
5

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

DirectSOFT 5



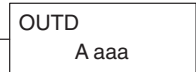
Handheld Programmer Keystrokes



### Out Double (OUTD)

DS5	Used
HPP	Used

The Out Double instruction is a 32 bit instruction that copies the value in the accumulator to two consecutive V memory locations at a specified starting location (Aaaa).

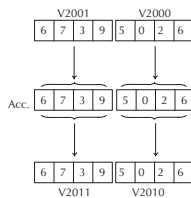
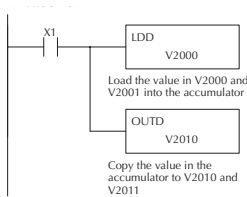


Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	All (See page 4-28)
Pointer ..... P	All V-memory (See page 4-28)

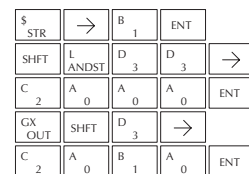
Discrete Bit Flags	Description
SP53	On when the pointer is outside of the available range.

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.

DirectSOFT 5



Handheld Programmer Keystrokes



### Out Formatted (OUTF)

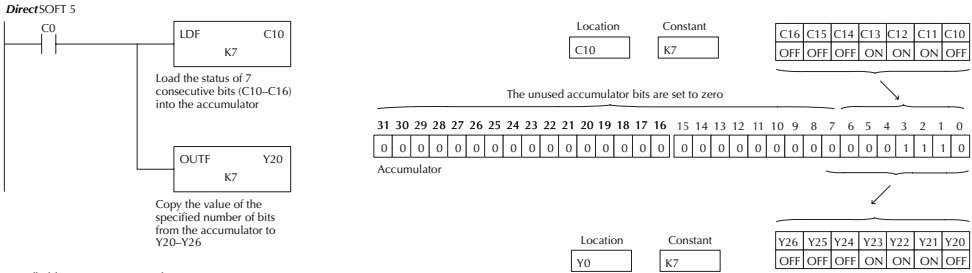
DS5	Used
HPP	Used

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.

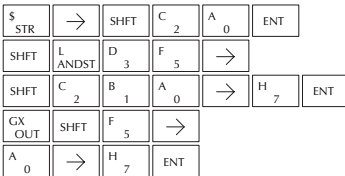


Operand Data Type		DL05 Range	
.....A		<b>aaa</b>	<b>bbb</b>
Inputs .....	X	0–377	—
Outputs .....	Y	0–377	—
Control Relays .....	C	0–777	—
Constant .....	K	—	1–32

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.



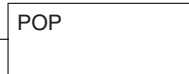
Handheld Programmer Keystrokes



### POP

DS5	Used
HPP	Used

The POP instruction moves the value from the first level of the accumulator stack (32 bits) to the accumulator and shifts each value in the stack up one level.

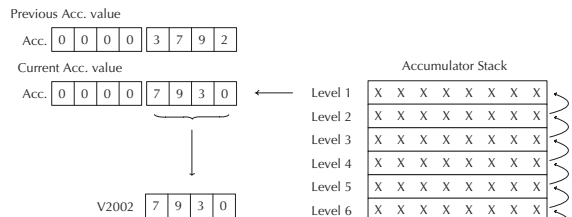
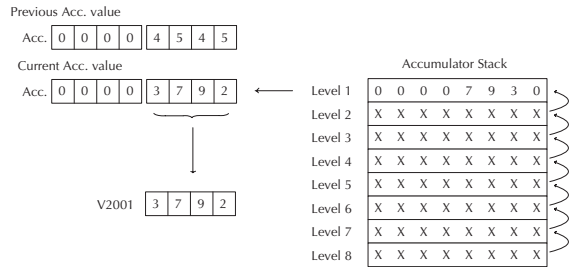
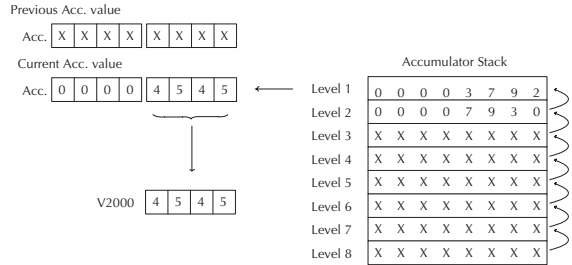
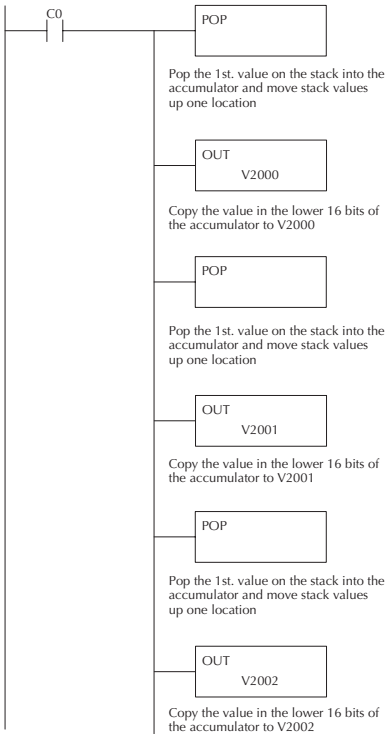


Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.

### Pop Instruction (cont'd)

In the example below, when C0 is on, the value 4545 that was on top of the stack is moved into the accumulator using the Pop instruction. The value is output to V2000 using the Out instruction. The next Pop moves the value 3792 into the accumulator and outputs the value to V2001. The last Pop moves the value 7930 into the accumulator and outputs the value to V2002. Please note if the value in the stack were greater than 16 bits (4 digits) the Out Double instruction would be used and two V-memory locations for each Out Double must be allocated.

DirectSOFT 5



Handheld Programmer Keystrokes

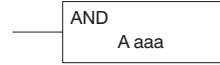
\$ STR	→	SHFT	C 2	A 0	ENT				
SHFT	P CV	SHFT	O INST#	P CV	ENT				
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	A 0	ENT	
SHFT	P CV	SHFT	O INST#	P CV	ENT				
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	B 1	ENT	
SHFT	P CV	SHFT	O INST#	P CV	ENT				
GX OUT	→	SHFT	V AND	C 2	A 0	A 0	C 2	ENT	

# Logical Instructions (Accumulator)

## And (AND)

DS5	Used
HPP	Used

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	DL05 Range
..... A	aaa
V-memory .....	V
Pointer .....	P

Discrete Bit Flags	Description
SP63	On if the result in the accumulator is zero.
SP70	On anytime the value in the accumulator is negative.

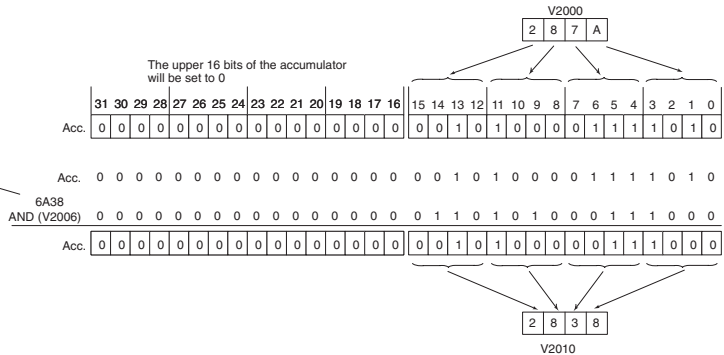
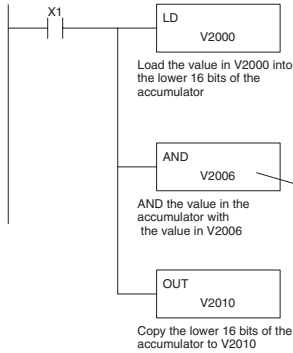
5



**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.

DirectSOFT 5



Handheld Programmer Keystrokes

\$	STR	→	B	1	ENT									
SHFT	L	D	3	→	C	2	A	0	A	0	A	0	ENT	
V	AND	→	SHFT	V	AND	C	2	A	0	A	0	G	6	ENT
GX	OUT	→	SHFT	V	AND	C	2	A	0	B	1	A	0	ENT

### And Double (ANDD)

DS5	Used
HPP	Used

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).

ANDD  
K aaa

Operand Data Type	DL05 Range
	<b>aaa</b>
V-memory .....	V
Pointer .....	P
Constant .....	K
	0-FFFFFFF

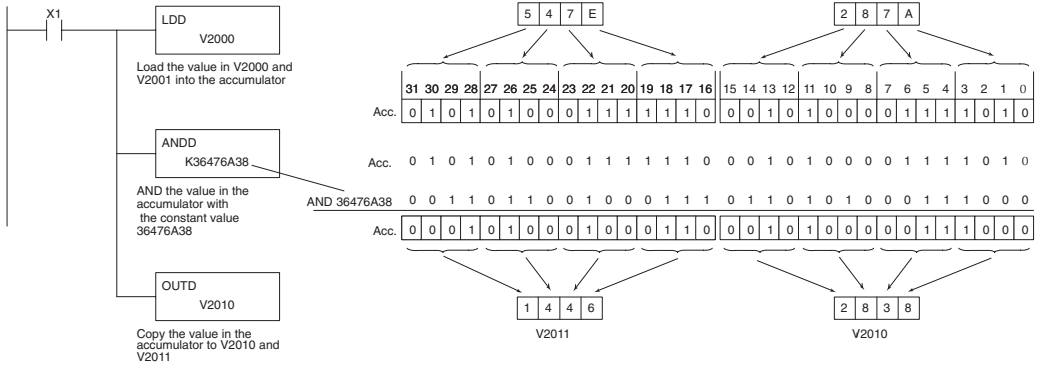
Discrete Bit Flags	Description
SP63	On if the result in the accumulator is zero.
SP70	On anytime the value in the accumulator is negative



**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 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.

DirectSOFT 5



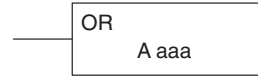
Handheld Programmer Keystrokes

\$	STR	→	B	1	ENT																						
SHFT	L	ANDST	D	3	D	3	→	C	2	A	0	A	0	A	0	A	0	ENT									
V	AND	SHFT	D	3	→	SHFT	K	JMP	D	3	G	6	E	4	H	7	G	6	SHFT	A	0	SHFT	D	3	I	8	ENT
GX	OUT	SHFT	D	3	→	C	2	A	0	B	1	A	0	ENT													

### Or (OR)

DS5	Used
HPP	Used

The Or instruction is a 16 bit instruction that logically ors the value in the lower 16 bits of the accumulator with a specified V-memory location (Aaaa). The result resides in the accumulator. The discrete status flag indicates if the result of the Or is zero.



Operand Data Type	DL05 Range
..... A	aaa
V-memory .....	V
Pointer .....	P

Discrete Bit Flags	Description
SP63	On if the result in the accumulator is zero.
SP70	On anytime the value in the accumulator is negative.

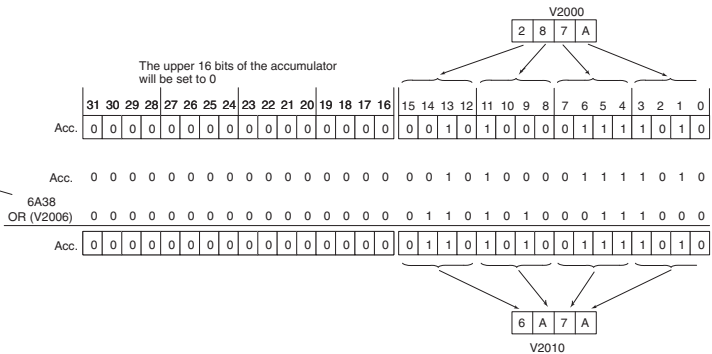
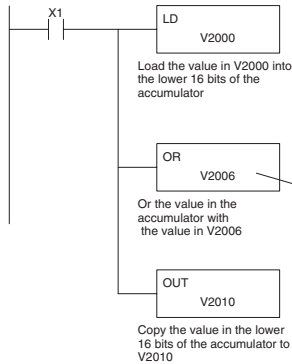
5



**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 ored with V2006 using the Or instruction. The value in the lower 16 bits of the accumulator are output to V2010 using the Out instruction.

DirectSOFT 5



Handheld Programmer Keystrokes

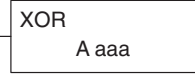
S	STR	→	B	1	ENT				
SHFT	L	ANDST	D	3	→	C	A	A	A
						2	0	0	0
Q	OR	→	SHFT	V	AND	C	A	A	G
						2	0	0	6
GX	OUT	→	SHFT	V	AND	C	A	B	A
						2	0	1	0



### Exclusive Or (XOR)

DS5	Used
HPP	Used

The Exclusive Or instruction is a 16 bit instruction that performs an exclusive or of the value in the lower 16 bits of the accumulator and a specified V-memory location (Aaaa). The result resides in the in the accumulator. The discrete status flag indicates if the result of the XOR is zero.



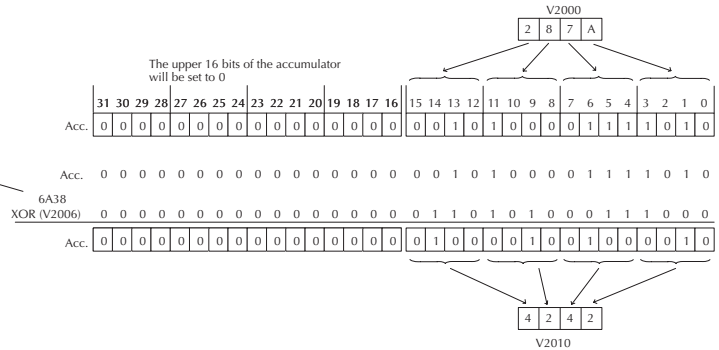
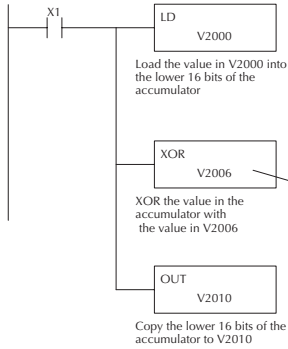
Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map

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

DirectSOFT 5



Handheld Programmer Keystrokes

\$ STR	→	SHFT	X SET	B 1	ENT						
SHFT	L ANDST	D 3	→	SHFT	V AND						
SHFT	X SET	SHFT	Q OR	→	SHFT	V AND	C 2	A 0	A 0	A 0	ENT
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT			



### Exclusive Or Double (XORD)

DS5	Used
HPP	Used

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



Operand Data Type	DL05 Range
..... A	aaa
V-memory .....	V
Pointer .....	P
Constant .....	K
	0-FFFFFFF

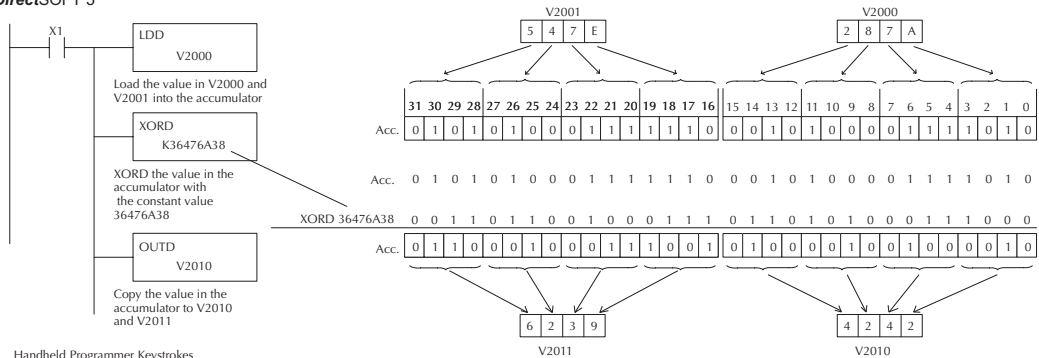
Discrete Bit Flags	Description
SP63	On if the result in the accumulator is zero.
SP70	On anytime the value in the accumulator is negative.



**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 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.

DirectSOFT 5



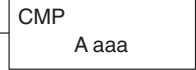
Handheld Programmer Keystrokes

\$ 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
CX OUT	SHFT	D 3	→ C 2 A 0 B 1 A 0 ENT

### Compare (CMP)

DS5	Used
HPP	Used

The compare instruction is a 16 bit instruction that compares the value in the lower 16 bits of the accumulator with the value in a specified V-memory location (Aaaa). The corresponding status flag will be turned on indicating the result of the comparison.

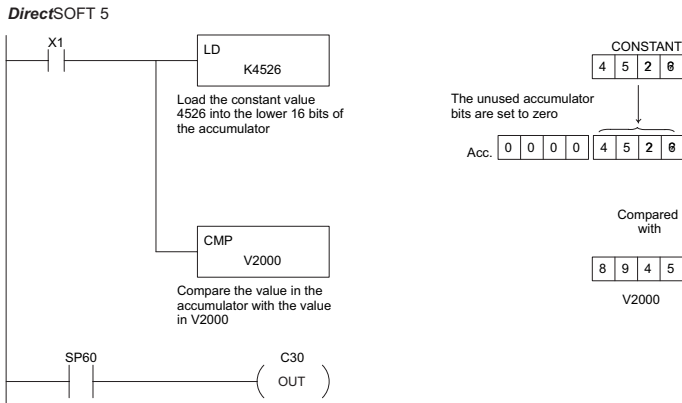


Operand Data Type	DL05 Range
..... A	aaa
V-memory .....	V
Pointer .....	P
	See memory map
	See memory map

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

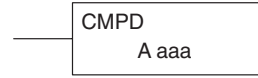
\$ STR	→	B 1	ENT											
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	F 5	C 2	G 6	ENT				
SHFT	C 2	SHFT	M ORST	P CV	→	C 2	A 0	A 0	A 0	ENT				
\$ STR	→	SHFT	SP STRN	G 6	A 0	ENT								
GX OUT	→	SHFT	C 2	D 3	A 0	ENT								



### 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.

DS5	Used
HPP	Used



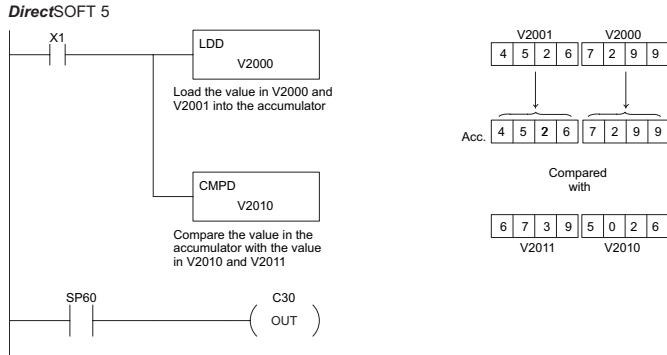
Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0-FFFFFFF

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

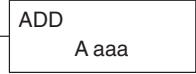
\$ STR	→	B 1	ENT								
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT		
SHFT	C 2	SHFT	M ORST	P CV	D 3	→	C 2	A 0	B 1	A 0	ENT
\$ STR	→	SHFT	SP STRN	G 6	A 0	ENT					
GX OUT	→	SHFT	C 2	D 3	A 0	ENT					

# Math Instructions

## Add (ADD)

DS5	Used
HPP	Used

Add is a 16 bit instruction that adds a BCD value in the accumulator with a BCD value in a V-memory location (Aaaa). The result resides in the accumulator.



Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16 bit addition instruction results in a carry.
SP67	On when the 32 bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

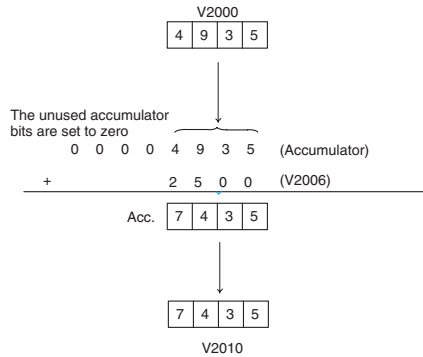
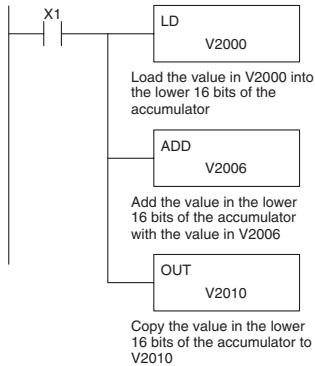
5



**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.

**DirectSOFT**



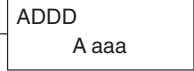
**Handheld Programmer Keystrokes**

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT	
SHFT	A 0	D 3	D 3	→	C 2	A 0	A 0	G 6	ENT
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT	

### Add Double (ADDD)

DS5	Used
HPP	Used

Add Double is a 32 bit instruction that adds the BCD value in the accumulator with a BCD value (Aaaa), which is either two consecutive V-memory locations or an 8-digit (max.) BCD constant. The result resides in the accumulator.



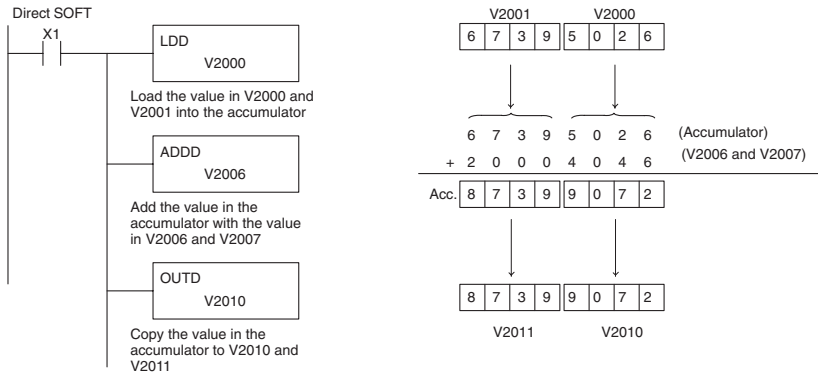
Operand Data Type	DL05 Range
..... A	<b>aaa</b>
V-memory .....	See memory map
Pointer .....	See memory map
Constant .....	0-99999999

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
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						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	A 0	D 3	D 3	D 3	→	C 2	A 0	A 0	G 6 ENT
GX OUT	SHFT	D 3	→	SHFT	V AND	C 2	A 0	B 1	A 0 ENT

### Subtract (SUB)

DS5	Used
HPP	Used

Subtract is a 16 bit instruction that subtracts the BCD value (Aaaa) in a V-memory location from the BCD value in the lower 16 bits of the accumulator. The result resides in the accumulator.

SUB
A aaa

Operand Data Type	DL05 Range
..... A	aaa
V-memory .....	V
Pointer .....	P
	See memory map
	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 addition instruction results in a borrow
SP65	On when the 32 bit addition 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.

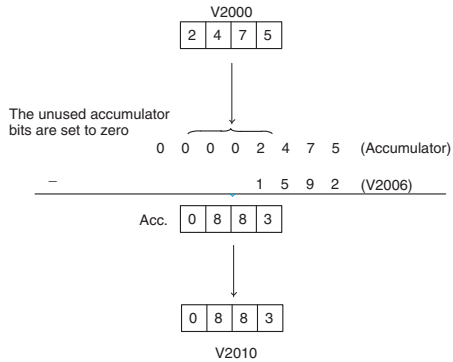
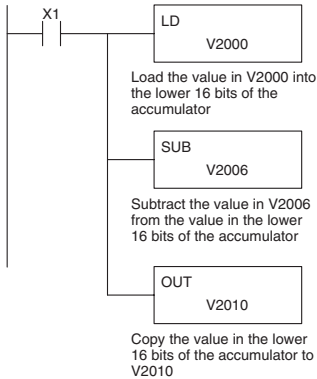
5



**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.

DirectSOFT



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT	
SHFT	S RST	U ISG	B 1	→	SHFT	V AND	C 2	A 0	A 0 G 6 ENT
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT	

### Subtract Double (SUBD)

DS5	Used
HPP	Used

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	Aaaa
------	------

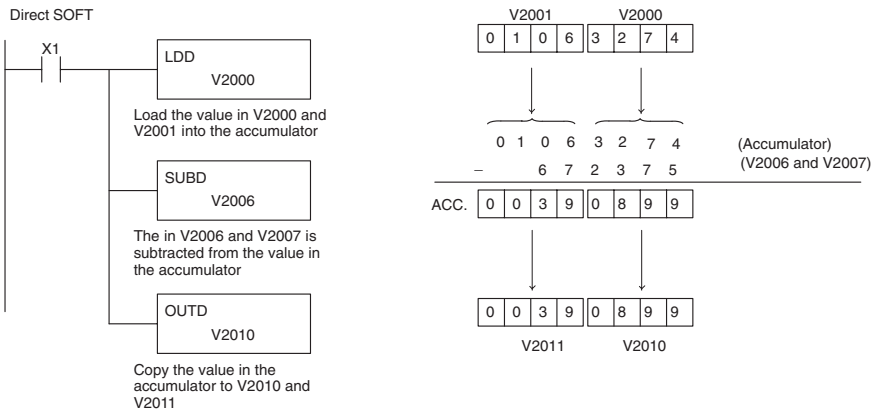
Operand Data Type	DL05 Range
..... A	<b>aaa</b>
V-memory .....	See memory map
Pointer .....	See memory map
Constant .....	0-99999999

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 addition instruction results in a borrow
SP65	On when the 32 bit addition 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.



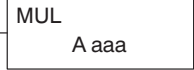
Handheld Programmer Keystrokes

\$	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					

### Multiply (MUL)

DS5	Used
HPP	Used

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.

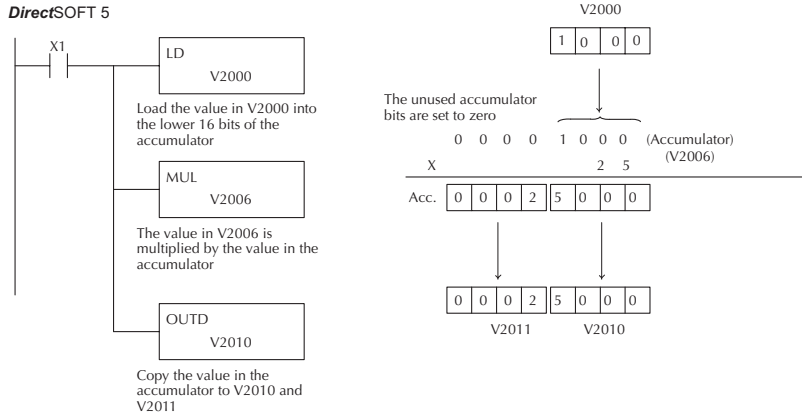


Operand Data Type	DL05 Range
..... A	<b>aaa</b>
V-memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0-9999

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

**NOTE:** 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)

DS5	Used
HPP	Used

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	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map

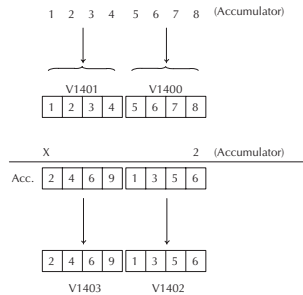
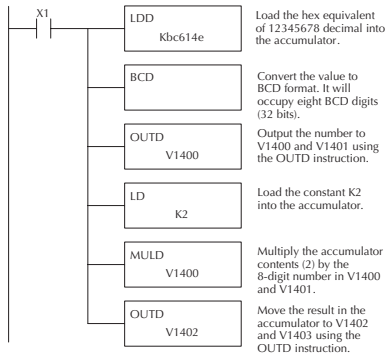
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.



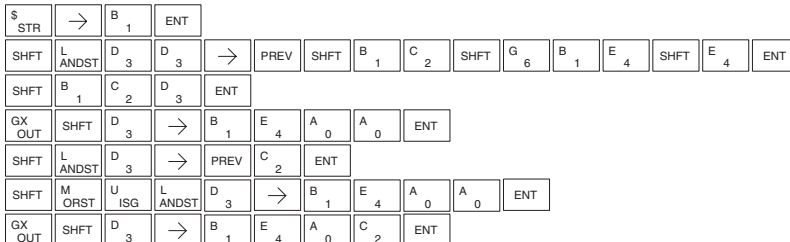
**NOTE:** Status flags are valid only until another instruction uses the same flag.

In the following example, when X1 is on, the constant Kbc614e hex will be loaded into the accumulator. When converted to BCD the number is "12345678". That number is stored in V1400 and V1401. After loading the constant K2 into the accumulator, we multiply it times 12345678, which is 24691356.

**DirectSOFT 5**



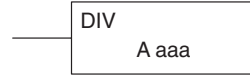
**Handheld Programmer Keystrokes**



### Divide (DIV)

DS5	Used
HPP	Used

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.

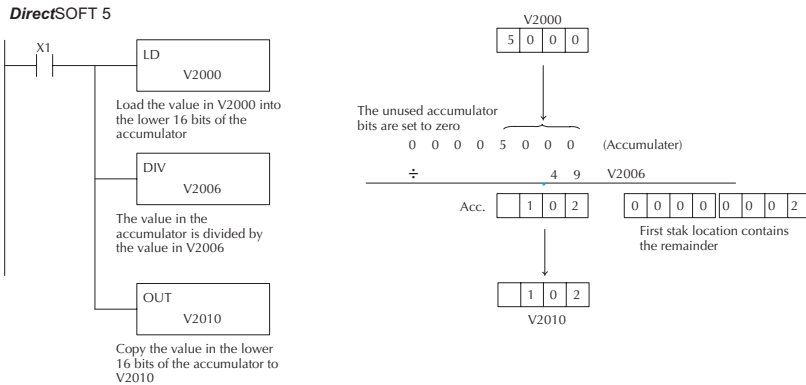


Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0-9999

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

**NOTE:** 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.



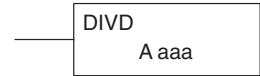
Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT					
SHIFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT
SHIFT	D 3	I 8	V AND	→	C 2	A 0	A 0	G 6 ENT
GX OUT	→	SHIFT	V AND	C 2	A 0	B 1	A 0	ENT

## Divide Double (DIVD)

DS5	Used
HPP	Used

Divide Double is a 32 bit instruction that divides the BCD value in the accumulator by a BCD value (Aaaa), which must be obtained from two consecutive V-memory locations. (You cannot use a constant as the parameter in the box.) The first part of the quotient resides in the accumulator and the remainder resides in the first stack location.



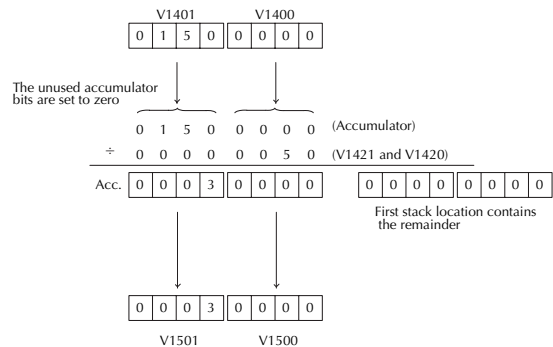
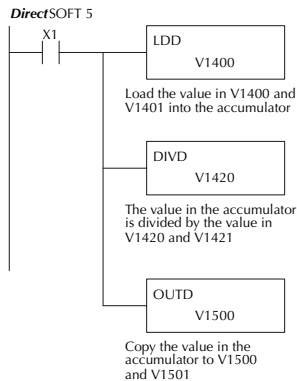
Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.



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

In the following example, when X1 is on, the value in V1400 and V1401 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is divided by the value in V1420 and V1421 using the Divide Double instruction. The first part of the quotient resides in the accumulator and the remainder resides in the first stack location. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



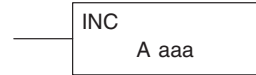
### Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT	SHFT	L ANDST	D 3	D 3	→
B 1	E 4	A 0	A 0	ENT	SHFT	D 3	I 8	V AND
→	B 1	E 4	C 2	A 0	ENT	GX OUT	SHFT	D 3
→	B 1	F 5	A 0	A 0	ENT			

### Increment (INC)

DS5	Used
HPP	Used

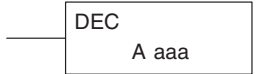
The Increment instruction increments a BCD value in a specified V-memory location by “1” each time the instruction is executed.



### Decrement (DEC)

DS5	Used
HPP	Used

The Decrement instruction decrements a BCD value in a specified V-memory location by “1” each time the instruction is executed.



5

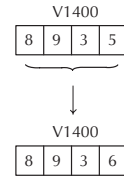
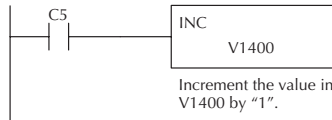
Operand Data Type	DL05 Range
..... A	aaa
V-memory .....	V
Pointer .....	P

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.

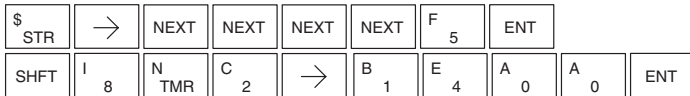


**NOTE:** Status flags are valid only until another instruction uses the same flag.

DirectSOFT 5

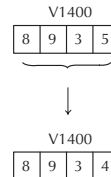
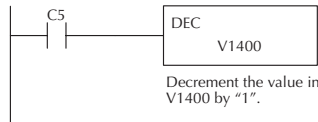


Handheld Programmer Keystrokes

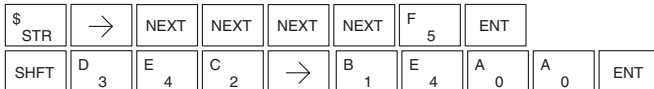


In the following increment example, when C5 is on the value in V1400 increases by one.

DirectSOFT 5



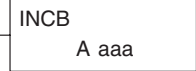
Handheld Programmer Keystrokes



### Increment Binary (INCB)

DS5	Used
HPP	Used

The Increment Binary instruction increments a binary value in a specified V-memory location by “1” each time the instruction is executed.

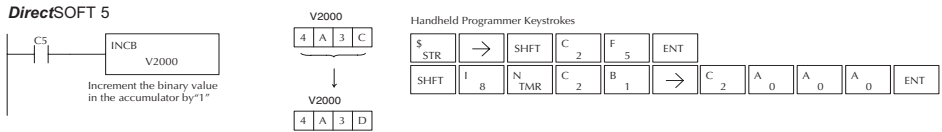


Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map

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

**5**

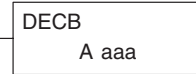
In the following example when C5 is on, the binary value in V2000 is increased by 1.



### Decrement Binary (DECB)

DS5	Used
HPP	Used

The Decrement Binary instruction decrements a binary value in a specified V-memory location by “1” each time the instruction is executed.



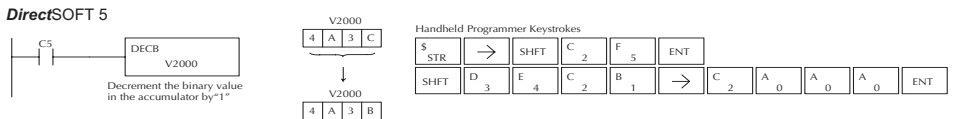
Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map

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 Binary (ADDB)

DS5	Used
HPP	Used

Add Binary is a 16 bit instruction that adds the binary value in the lower 16 bits of the accumulator with a binary value (Aaaa), which is either a V-memory location or a 16-bit constant. The result can be up to 32 bits and resides in the accumulator.

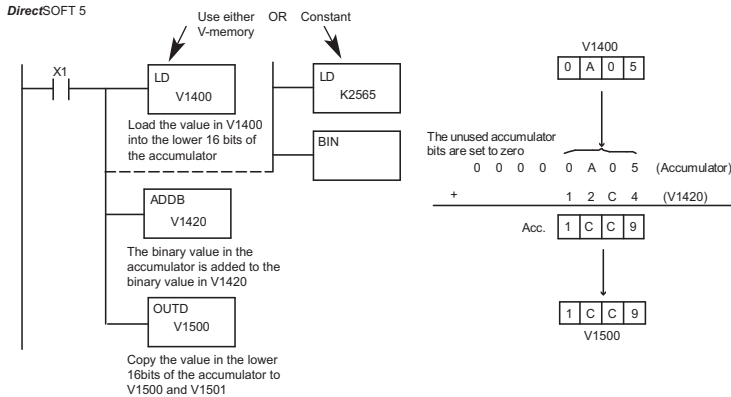


Operand Data Type	DL05 Range
..... A	<b>aaa</b>
V-memory .....	V
Pointer .....	P
Constant .....	K
	0-FFFF

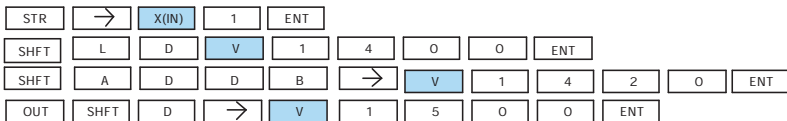
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP66	On when the 16 bit addition instruction results in a carry.
SP67	On when the 32 bit addition instruction results in a carry.
SP70	On anytime the value in the accumulator is negative.
SP73	On when a signed addition or subtraction results in a incorrect sign bit.

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

In the following example, when X1 is on, the value in V1400 will be loaded into the accumulator using the Load instruction. The binary value in the accumulator will be added to the binary value in V1420 using the Add Binary instruction. The value in the accumulator is copied to V1500 and V1501 using the Out Double instruction.



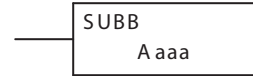
Handheld Programmer Keystrokes



### Subtract Binary (SUBB)

DS5	Used
HPP	Used

Subtract Binary is a 16 bit instruction that subtracts the binary value (Aaaa), which is either a V-memory location or a 4-digit (max.) binary constant, from the binary value in the accumulator. The result resides in the accumulator.



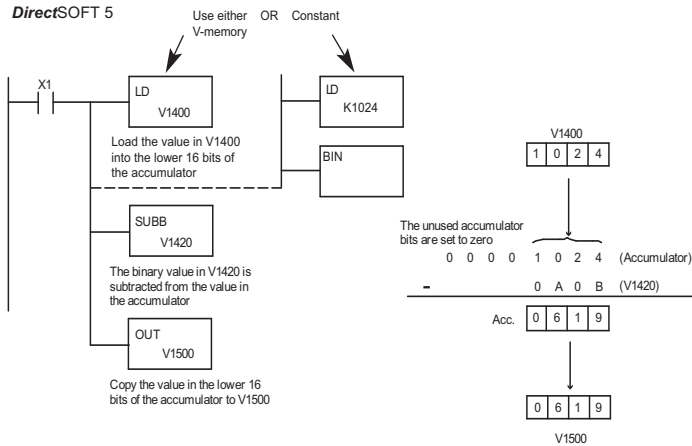
Operand Data Type	DL05 Range
..... A	<b>aaa</b>
V-memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0-FFFF

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 addition instruction results in a borrow
SP65	On when the 32 bit addition 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 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.



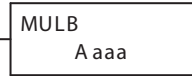
Handheld Programmer Keystrokes

STR	→	X(IN)	1	ENT					
SHFT	L	D	V	1	4	0	0	ENT	
SHFT	S	SHFT	U	B	B	→			
V	1	4	2	0	ENT				
OUT	SHFT	D	→	V	1	5	0	0	ENT

### Multiply Binary (MULB)

DS5	Used
HPP	Used

Multiply Binary is a 16 bit instruction that multiplies the binary value (Aaaa), which is either a V-memory location or a 4-digit (max.) binary constant, by the binary value in the accumulator. The result can be up to 32 bits and resides in the accumulator.

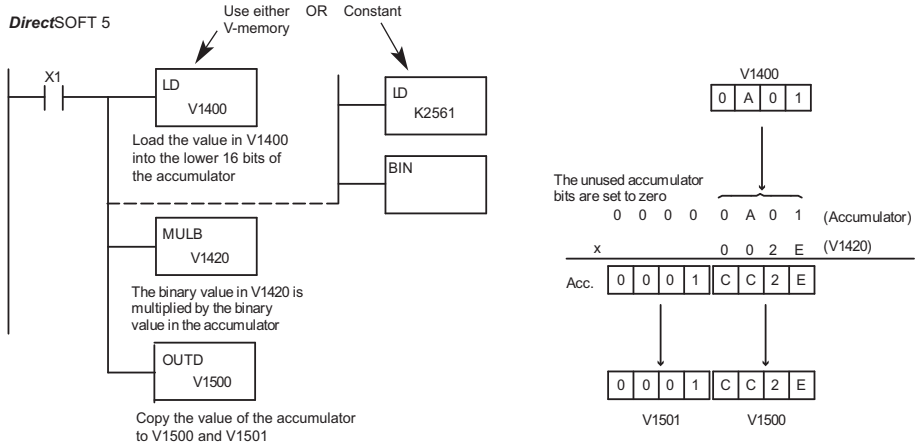


Operand Data Type	DL05 Range
..... A	<b>aaa</b>
V-memory .....	V See memory map
Pointer .....	P See memory map
Constant .....	K 1-FFFF

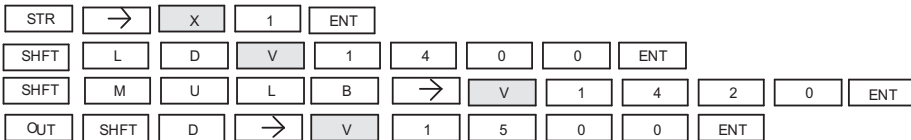
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On any time the value in the accumulator is negative.

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

In the following example, when X1 is on, the 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 and V1501 using the Out Double instruction.



Handheld Programmer Keystrokes

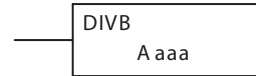




### Divide Binary (DIVB)

DS5	Used
HPP	Used

Divide Binary is a 16 bit instruction that divides the binary value in the accumulator by a binary value (Aaaa), which is either a V-memory location or a 16-bit (max.) binary constant. The first part of the quotient resides in the accumulator and the remainder resides in the first stack location.



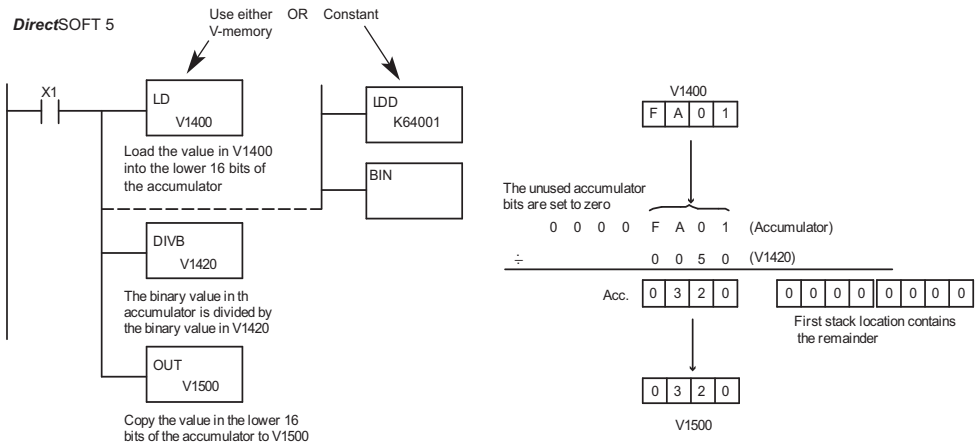
Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map
Constant ..... K	0–FFFF

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.



Handheld Programmer Keystrokes

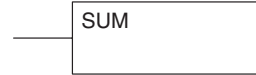
STR	→	X	1	ENT								
SHFT	L	D	V	1	4	0	0	ENT				
SHFT	D	I	V	B	→	V	1	4	2	0	ENT	
OUT	SHFT	D	→	V	1	5	0	0	ENT			

# Bit Operation Instructions

## Sum (SUM)

DS5	Used
HPP	Used

The Sum instruction counts number of bits that are set to “1” in the accumulator. The HEX result resides in the accumulator.

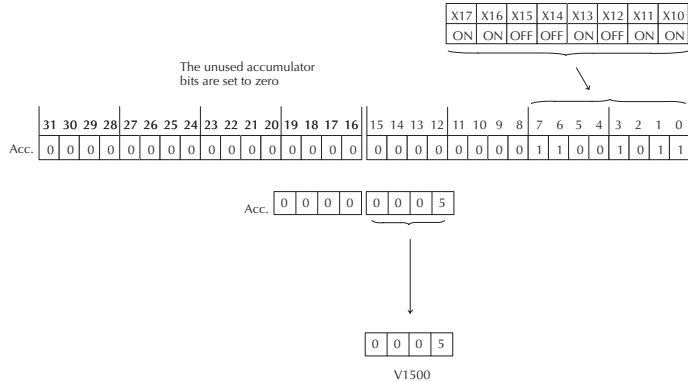
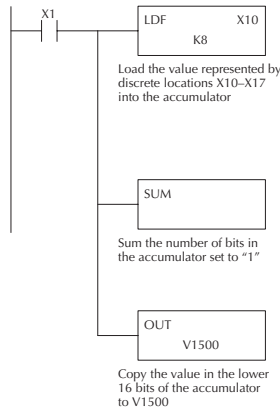


Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.

5

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.

**DirectSOFT 5**



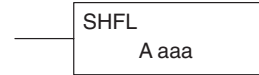
**Handheld Programmer Keystrokes**

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	F 5	→	B 1	A 0	→	I 8	ENT
SHFT	S RST	SHFT	U ISG	M ORST	→	ENT			
GX OUT	→	PREV	PREV	PREV	B 1	F 5	A 0	A 0	ENT

### Shift Left (SHFL)

DS5	Used
HPP	Used

Shift Left is a 32 bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the left. The vacant positions are filled with zeros and the bits shifted out of the accumulator are discarded.



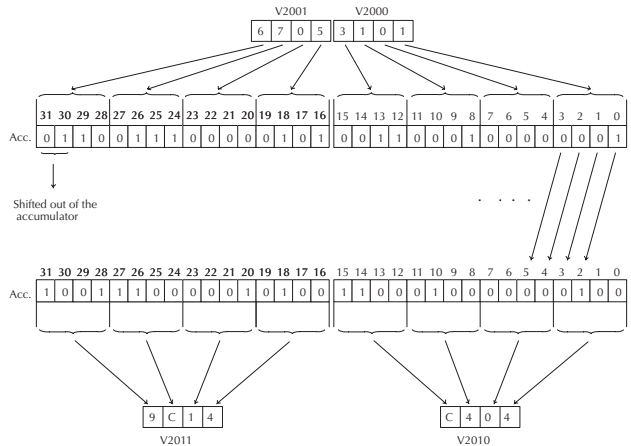
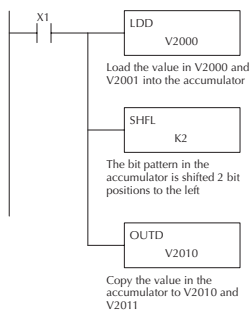
Operand Data Type	DL05 Range
..... A	<b>aaa</b>
V-memory ..... V	See memory map
Constant ..... K	1-32

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.

5

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.

**DirectSOFT 5**



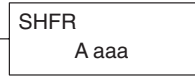
**Handheld Programmer Keystrokes**

\$ 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)

DS5	Used
HPP	Used

Shift Right is a 32 bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the right. The vacant positions are filled with zeros and the bits shifted out of the accumulator are lost.

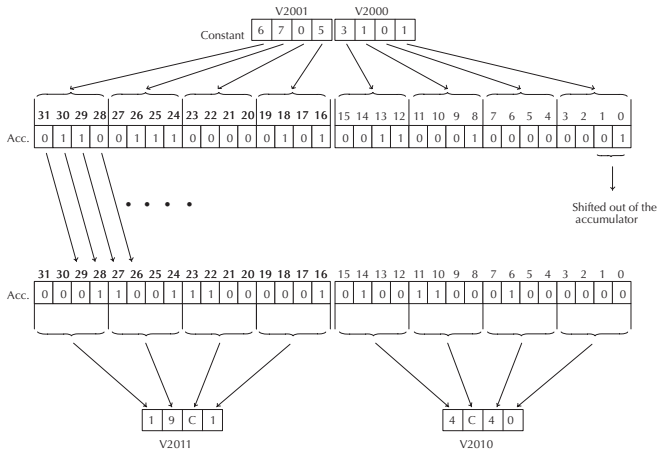
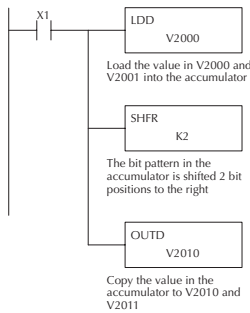


Operand Data Type	DL05 Range
..... A	<b>aaa</b>
V-memory ..... V	See memory map
Constant ..... K	1-32

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 right using the Shift Right instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.

DirectSOFT 5



Handheld Programmer Keystrokes

\$	→	B	1	ENT											
STR		D	3	D	3	→	C	2	A	0	A	0	A	0	ENT
SHFT	L	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	

### Encode (ENCO)

DS5	Used
HPP	Used

The Encode instruction encodes the bit position in the accumulator having a value of 1, and returns the appropriate binary representation. If the most significant bit is set to 1 (Bit 31), the Encode instruction would place the value HEX 1F (decimal 31) in the accumulator. If the value to be encoded is 0000 or 0001, the instruction will place a zero in the accumulator. If the value to be encoded has more than one bit position set to a “1”, the least significant “1” will be encoded and SP53 will be set on.



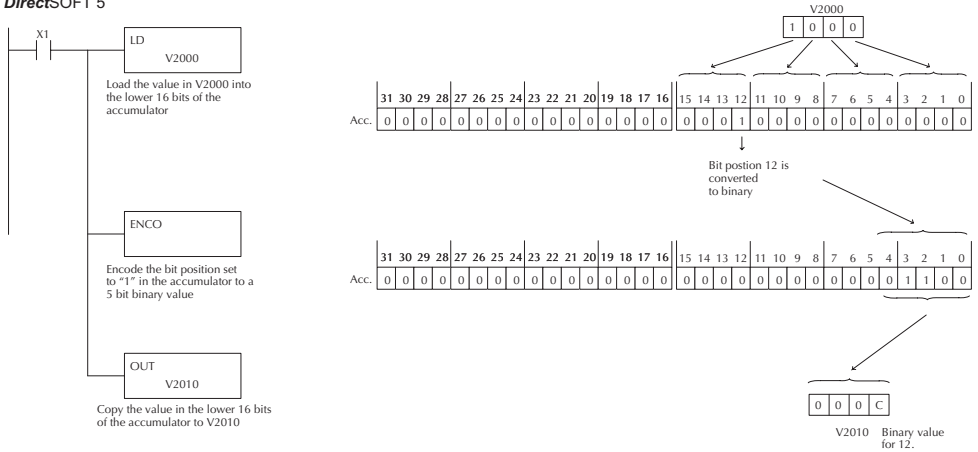
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP70	On anytime the value in the accumulator is negative.



**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.

DirectSOFT 5



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)

DS5	Used
HPP	Used

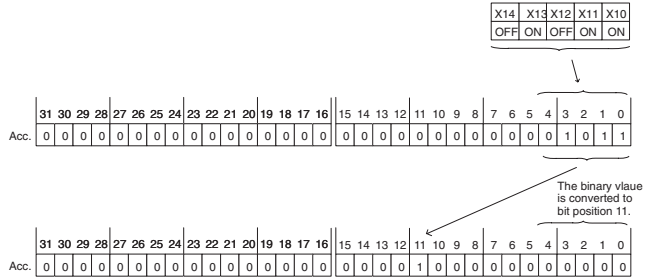
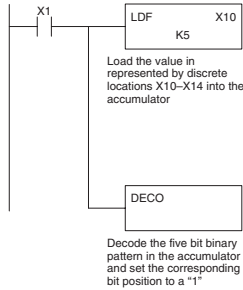
The Decode instruction decodes a 5 bit binary value of 0–31 (0–1F HEX) in the accumulator by setting the appropriate bit position to a 1. If the accumulator contains the value F (HEX), bit 15 will be set in the accumulator. If the value to be decoded is greater than 31, the number is divided by 32 until the value is less than 32 and then the value is decoded.



In the following example when X1 is on, the value formed by discrete locations X10–X14 is loaded into the accumulator using the Load Formatted instruction. The five bit binary pattern in the accumulator is decoded by setting the corresponding bit position to a “1” using the Decode instruction.

5

DirectSOFT 5



Handheld Programmer Keystrokes

\$	STR	→	B	1	ENT										
SHFT	L	ANDST	D	3	F	5	→	B	1	A	0	→	F	5	ENT
SHFT	D	3	E	4	C	2	O	INST#	ENT						

# Number Conversion Instructions (Accumulator)

## Binary (BIN)

DS5	Used
HPP	Used

The Binary instruction converts a BCD value in the accumulator to the equivalent binary value. The result resides in the accumulator.

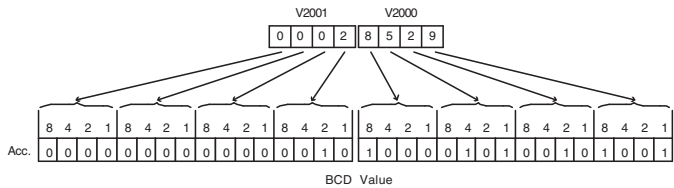
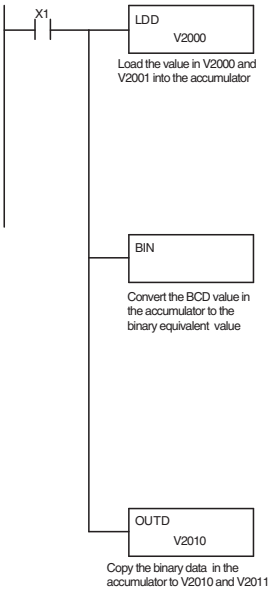


In the following example, when X1 is on, the value in V2000 and V2001 is loaded into the accumulator using the Load Double instruction. The BCD value in the accumulator is converted to the binary (HEX) equivalent using the BIN instruction. The binary value in the accumulator is copied to V2010 and V2011 using the Out Double instruction. (The handheld programmer will display the binary value in V2010 and V2011 as a HEX value.)

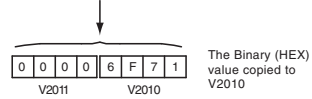
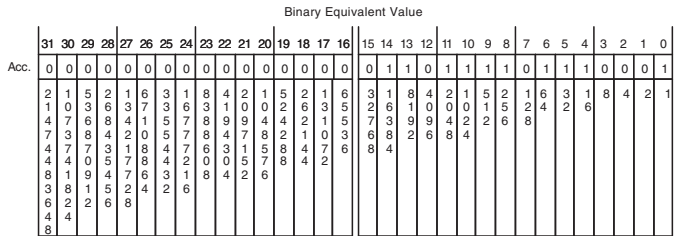
5

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.

DirectSOFT 5



$$28529 = 16384 + 8192 + 2048 + 1024 + 512 + 256 + 64 + 32 + 16 + 1$$



Handheld Programmer Keystrokes

\$	STR	→	B	1	ENT											
SHFT	L	ANDST	D	3	→	D	3	C	2	A	0	A	0	A	0	ENT
SHFT	B	1	I	8	→	N	TMR	ENT								
GX	OUT	SHFT	D	3	→	C	2	A	0	B	1	A	0	ENT		

### Binary Coded Decimal (BCD)

DS5	Used
HPP	Used

The Binary Coded Decimal instruction converts a binary value in the accumulator to the equivalent BCD value. The result resides in the accumulator.

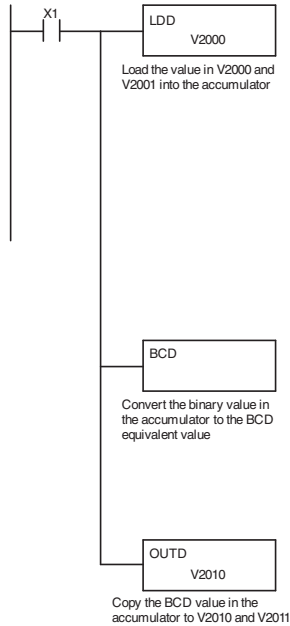


In the following example, when X1 is on, the binary (HEX) value in V2000 and V2001 is loaded into the accumulator using the Load Double instruction. The binary value in the accumulator is converted to the BCD equivalent value using the BCD instruction. The BCD value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.

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.

5

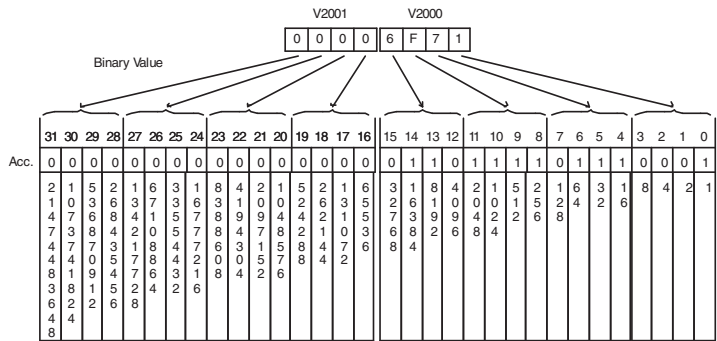
DirectSOFT 5



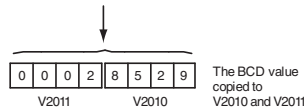
Load the value in V2000 and V2001 into the accumulator

Convert the binary value in the accumulator to the BCD equivalent value

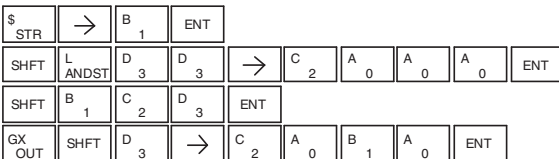
Copy the BCD value in the accumulator to V2010 and V2011



$$16384 + 8192 + 2048 + 1024 + 512 + 256 + 64 + 32 + 16 + 1 = 28529$$



Handheld Programmer Keystrokes

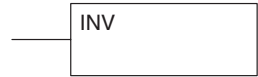




## Invert (INV)

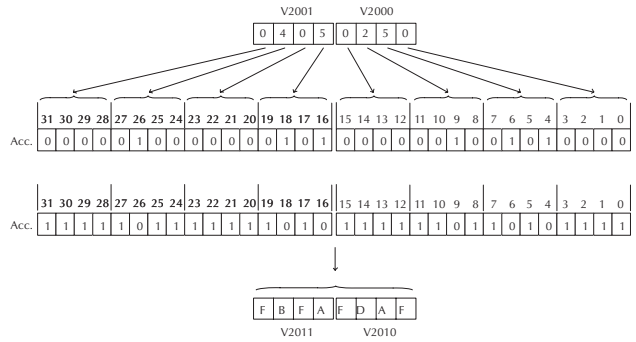
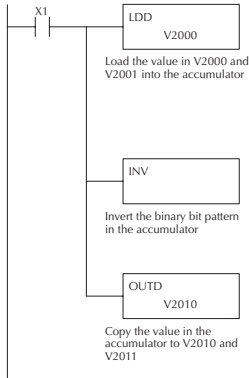
DS5	Used
HPP	Used

The Invert instruction inverts or takes the one's complement of the 32 bit value in the accumulator. The result resides in the accumulator.



In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is inverted using the Invert instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.

### DirectSOFT 5



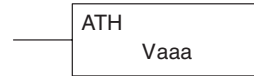
### Handheld Programmer Keystrokes

\$	STR	→	B	1	ENT											
SHFT	L	ANDST	D	3	D	3	→	C	2	A	0	A	0	A	0	ENT
SHFT	I	8	N	TMR	V	AND	ENT									
GX	OUT	SHFT	D	3	→	C	2	A	0	B	1	A	0		ENT	

### ASCII to HEX (ATH)

DS5	Used
HPP	Used

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.



5

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.

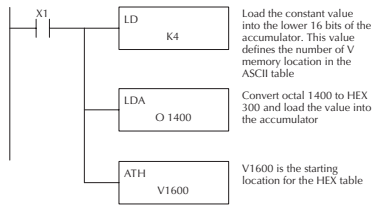
Operand Data Type	DL05 Range
	<b>aaa</b>
V-memory .....V	See memory map

Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.

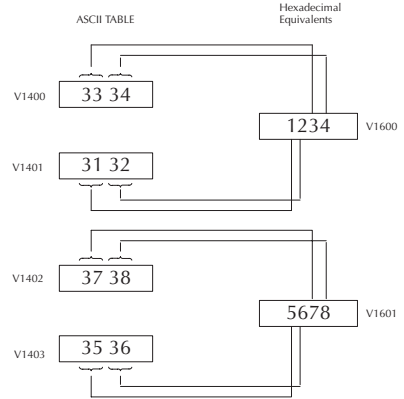
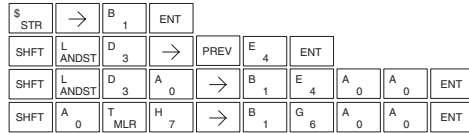
In the example on the following page, when X1 is ON the constant (K4) is loaded into the accumulator using the Load instruction and will be placed in the first level of the accumulator stack when the next Load instruction is executed. The starting location for the ASCII table (V1400) is loaded into the 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

## DirectSOFT 5



### Handheld Programmer Keystrokes



## HEX to ASCII (HTA)

DS5	Used
HPP	Used

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.

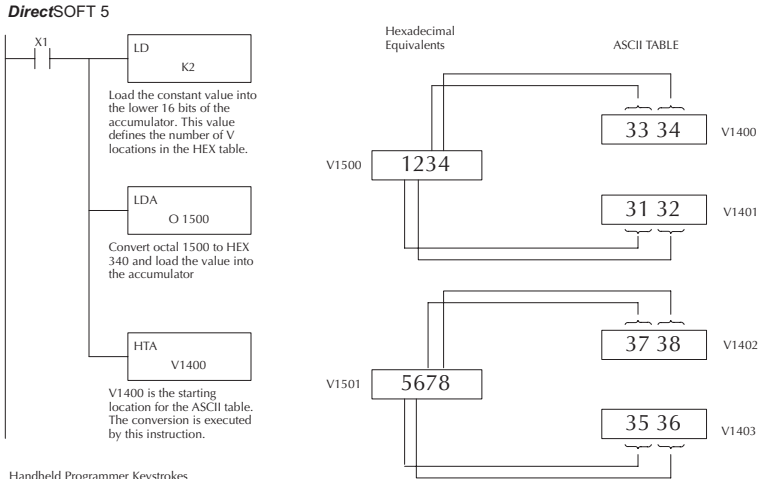
- Step 1: — Load the number of V-memory locations in the HEX table into the first level of the accumulator stack.
- Step 2: — Load the starting V-memory location for the HEX table into the accumulator. This parameter must be a HEX value.
- Step 3: — Specify the starting V-memory location (Vaaa) for the 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	DL05 Range
V-memory .....V	aaa 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.



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	ENT		
SHFT	L ANDST	D 3	A 0	→	B 1	F 5	A 0	A 0	ENT
SHFT	H 7	T MLR	A 0	→	B 1	E 4	A 0	A 0	ENT

The table below lists valid ASCII values for HTA conversion.

ASCII Values Valid for HTA Conversion			
Hex Value	ASCII Value	Hex Value	ASCII Value
0	30	8	38
1	31	9	39
2	32	A	41
3	33	B	42
4	34	C	43
5	35	D	44
6	36	E	45
7	37	F	46



### Shuffle Digits (SFLDGT)

DS5	Used
HPP	Used

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.

5



**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)

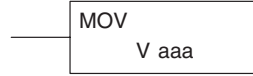


# Table Instructions

## Move (MOV)

DS5	Used
HPP	Used

The Move instruction moves the values from a V-memory table to another V-memory table the same length (a table is a consecutive group of V-memory locations). The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. The MOV instruction can be used to write data to non-volatile V-memory (see Appendix F). Listed below are the steps necessary to program the MOV function.



- Step 1:— Load the number of V-memory locations to be moved into the first level of the accumulator stack. This parameter is a HEX value (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.

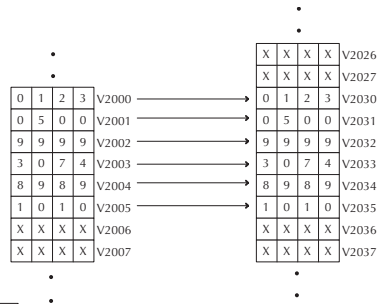
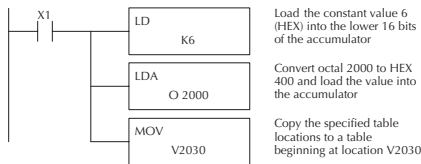
Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Pointer ..... P	See memory map

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.

**DirectSOFT 5**



**Handheld Programmer Keystrokes**

\$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	→	SHFT	K JMP	G 6	ENT		
SHFT	L ANDST	D 3	A 0	→	C 2	A 0	A 0	A 0	ENT
SHFT	M ORST	O INST#	V AND	→	C 2	A 0	D 3	A 0	ENT



## Move Memory Cartridge (MOVMC)

and

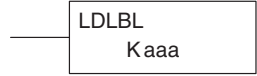
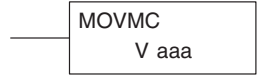
## Load Label (LDLBL)

DS5	Used
HPP	Used

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.



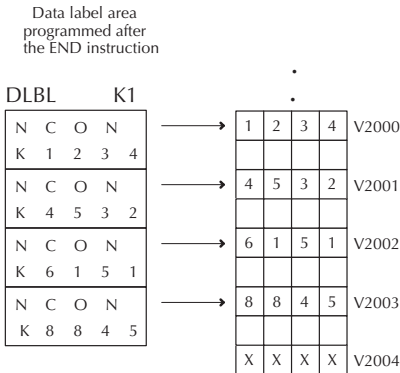
5

Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map

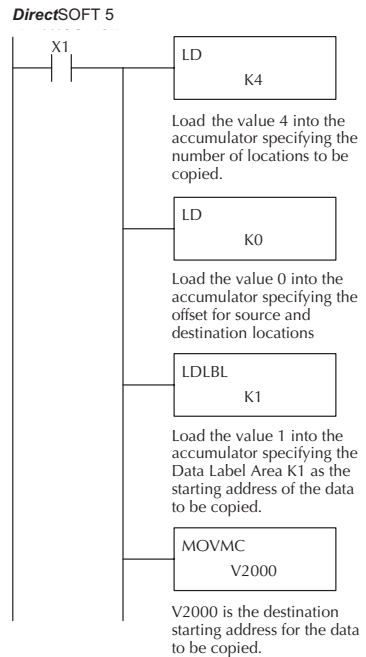
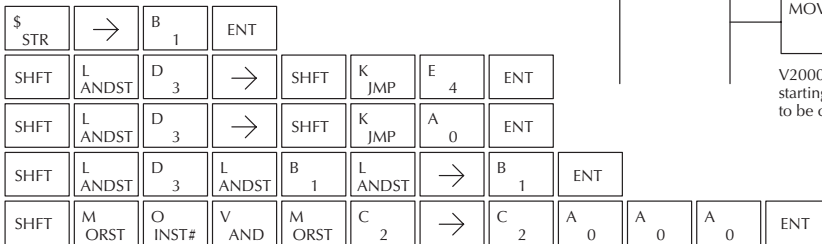
### Copy Data From a Data Label Area to V-memory

In the example to the right, 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.

5



Handheld Programmer Keystrokes



# CPU Control Instructions

## No Operation (NOP)

DS5	Used
HPP	N/A

The No Operation is an empty (not programmed) memory location.

—( NOP )



DirectSOFT 5

Handheld Programmer Keystrokes

SHFT	N TMR	O INST#	P CV	ENT
------	----------	------------	---------	-----

## End (END)

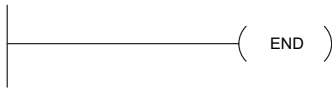
DS5	Used
HPP	N/A

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 )

DirectSOFT 5

Handheld Programmer Keystrokes



SHFT	E 4	N TMR	D 3	ENT
------	--------	----------	--------	-----

## Stop (STOP)

DS5	Used
HPP	N/A

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.

DirectSOFT 5

Handheld Programmer Keystrokes



\$ STR	→	SHFT	C 2	A 0	ENT	
SHFT	S RST	SHFT	T MLR	O INST#	P CV	ENT

Discrete Bit Flags	Description
SP16	On when the DL05 goes into the TERM_PRG mode.
SP53	On when the DL05 goes into the PRG mode.

### Reset Watch Dog Timer (RSTWT)

DS5	Used
HPP	N/A

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.

—(RSTWT)

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.

5

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 5



Handheld Programmer Keystrokes

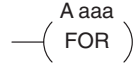
SHIFT	R ORN	S RST	T MLR	W ANDN	T MLR	ENT
-------	----------	----------	----------	-----------	----------	-----

# Program Control Instructions

## For / Next (FOR) (NEXT)

DS5	Used
HPP	Used

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.

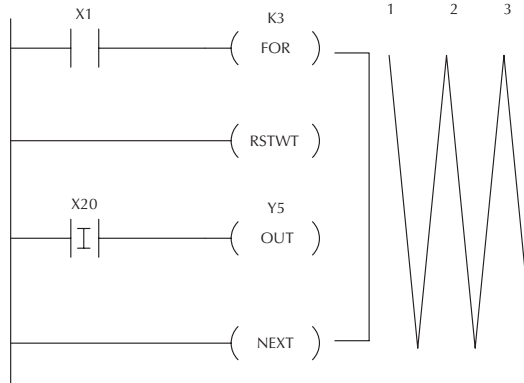


5

Operand Data Type	DL05 Range
..... A	aaa
V-memory .....	See memory map
Constant .....	1-9999

In the following example, when X1 is on, the application program inside the For / Next loop will be executed three times. If X1 is off the program inside the loop will not be executed. The immediate instructions may or may not be necessary depending on your application. Also, The RSTWT instruction is not necessary if the For / Next loop does not extend the scan time larger the Watch Dog Timer setting. For more information on the Watch Dog Timer, refer to the RSTWT instruction.

DirectSOFT 5



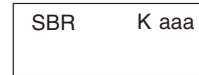
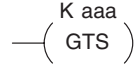
Handheld Programmer Keystrokes

\$ 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)

DS5	Used
HPP	Used

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 64 GTS instructions and 64 SBR instructions used in a program. The GTS instructions can be nested up to 8 levels. An error E412 will occur if the maximum limits are exceeded. Typically this will be used in an application where a block of program logic may be slow to execute and is not required to execute every scan. The subroutine label and all associated logic is placed after the End statement in the program. When the subroutine is called from the main program, the CPU will execute the subroutine (SBR) with the same constant number (K) as the GTS instruction which called the subroutine.



By placing code in a subroutine it is only scanned and executed when needed since it resides after the End instruction. Code which is not scanned does not impact the overall scan time of the program.

Operand Data Type	DL05 Range
..... A	aaa
Constant ..... K	1-FFFF

### Subroutine Return (RT)

DS5	Used
HPP	Used

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).



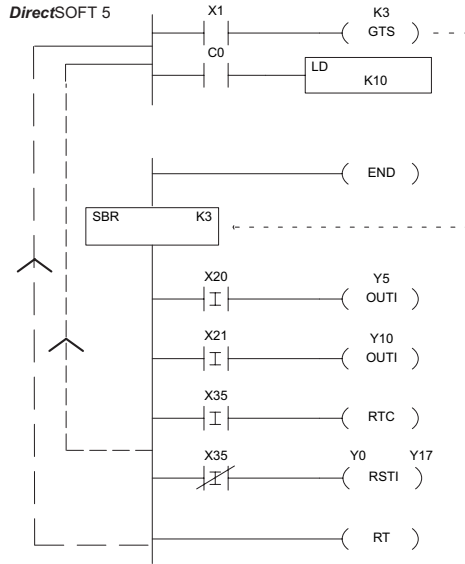
### Subroutine Return Conditional (RTC)

DS5	Used
HPP	Used

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.



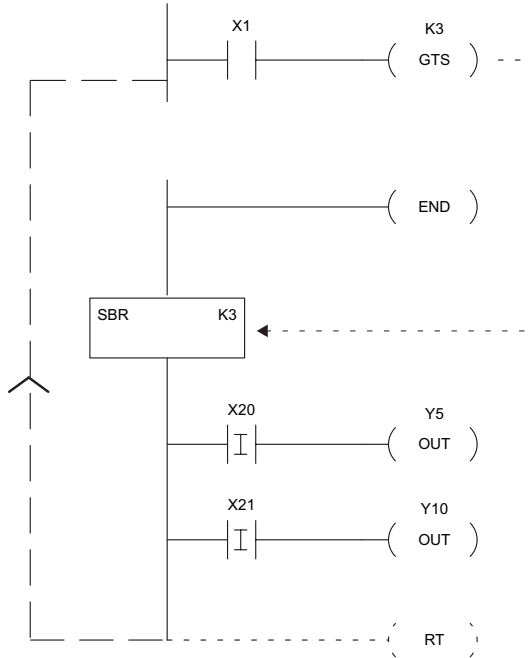
Handheld Programmer Keystrokes

\$ 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
SHFT	R ORN	T MLR	ENT				

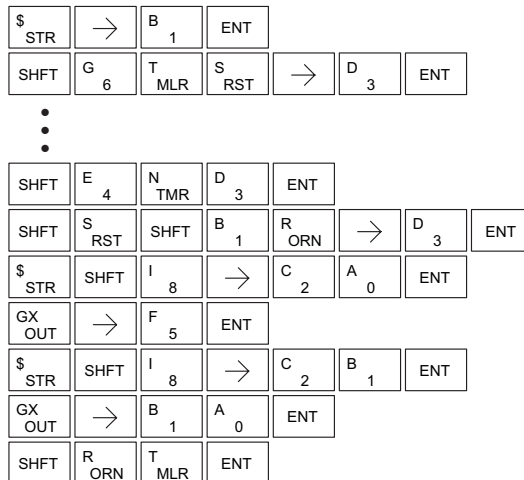


In the following example, when X1 is on, Subroutine K3 will be called. The CPU will jump to the Subroutine Label K3 and the ladder logic in the subroutine will be executed. The CPU will return to the main body of the program after the RT instruction is executed.

*DirectSOFT 5*



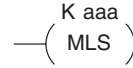
**Handheld Programmer Keystrokes**



### Master Line Set (MLS)

DS5	Used
HPP	Used

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.



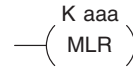
Operand Data Type	DL05 Range
.....A	aaa
Constant .....K	1-7

5

### Master Line Reset (MLR)

DS5	Used
HPP	Used

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.

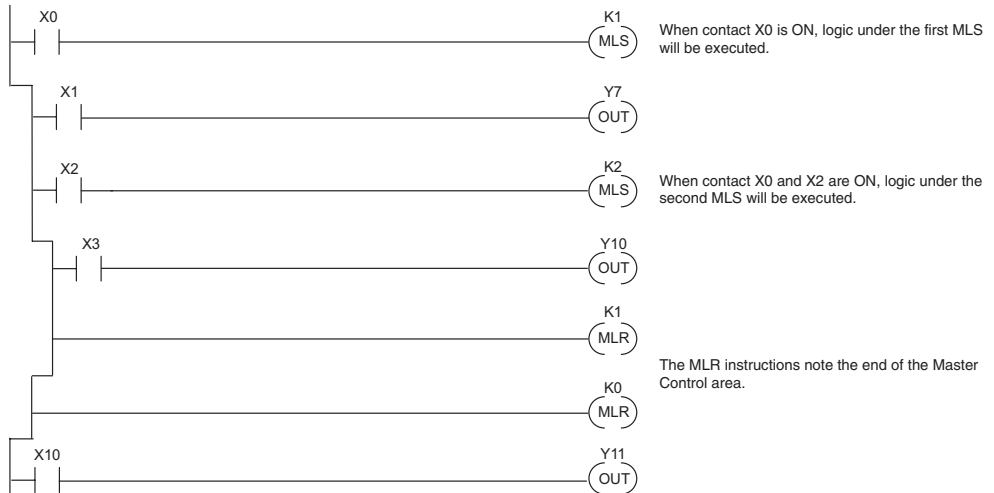


### Understanding Master Control Relays

Operand Data Type	DL05 Range
.....A	aaa
Constant .....K	0-7

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.

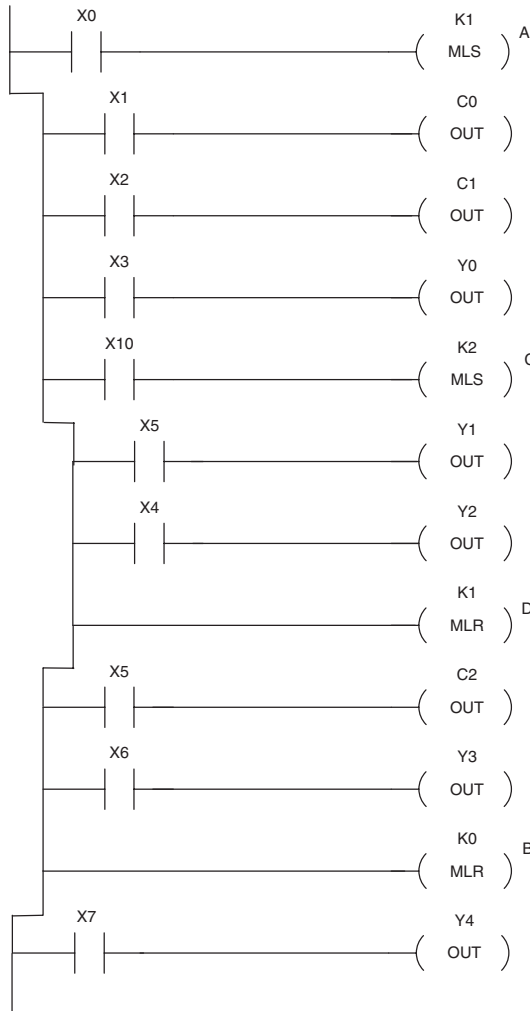
DirectSOFT 5



### 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.

DirectSOFT 5



Handheld Programmer Keystrokes

\$ STR	→	A 0	ENT		
Y MLS	→	B 1	ENT		
\$ STR	→	B 1	ENT		
GX OUT	→	SHFT	C 2	A 0	ENT
\$ STR	→	C 2	ENT		
GX OUT	→	SHFT	C 2	B 1	ENT
\$ STR	→	D 3	ENT		
GX OUT	→	A 0	ENT		
\$ STR	→	B 1	A 0	ENT	
Y MLS	→	C 2	ENT		
\$ STR	→	F 5	ENT		
GX OUT	→	B 1	ENT		
\$ STR	→	E 4	ENT		
GX OUT	→	C 2	ENT		
T MLR	→	B 1	ENT		
\$ STR	→	F 5	ENT		
GX OUT	→	SHFT	C 2	C 2	ENT
\$ STR	→	G 6	ENT		
GX OUT	→	D 3	ENT		
T MLR	→	A 0	ENT		
\$ STR	→	H 7	ENT		
GX OUT	→	E 4	C 2	ENT	

# Interrupt Instructions

## Interrupt (INT)

DS5	Used
HPP	Used

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 (5–999 mS).

INT	O aaa
-----	-------

INT	1
-----	---

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 DL05, only one hardware interrupt is available.

Operand Data Type	DL05 Range
Constant .....	0
	0, 1

## Interrupt Return (IRT)

DS5	Used
HPP	Used

An Interrupt Return is normally executed as the last instruction in the interrupt routine. It returns the CPU to the point in the main program from which it was called. The Interrupt Return is a stand-alone instruction (no input contact on the rung).



## Interrupt Return Conditional (IRTC)

DS5	Used
HPP	Used

The Interrupt Return Conditional instruction is an optional instruction used with an input contact to implement a conditional return from the interrupt routine. The Interrupt Return is required to terminate the interrupt routine.



## Enable Interrupts (ENI)

DS5	Used
HPP	Used

The Enable Interrupt instruction is placed in the main ladder program (before the End instruction), enabling the interrupt. The interrupt remains enabled until the program executes a Disable Interrupt instruction.



### Disable Interrupts (DISI)

DS5	Used
HPP	Used

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.

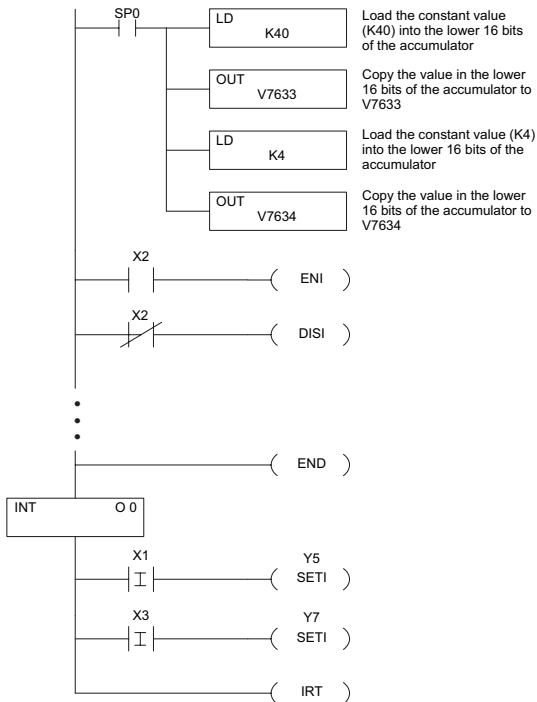
—( DISI )

### 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 0 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.

DirectSOFT



Handheld Programmer Keystrokes

\$ STR	→	SHFT	SP STRN	A 0	ENT				
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	A 0	ENT	
GX OUT	→	SHFT	V AND	H 7	G 6	D 3	D 3	ENT	
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	ENT		
GX OUT	→	SHFT	V AND	H 7	G 6	D 3	E 4	ENT	
\$ STR	→	C 2	ENT						
SHFT	E 4	N TMR	I 8	ENT					
SP STRN	→	C 2	ENT						
SHFT	D 3	I 8	S RST	I 8	ENT				
...									
SHFT	E 4	N TMR	D 3	ENT					
SHFT	I 8	N TMR	T MLR	→	A 0	ENT			
\$ STR	SHFT	I 8	→	B 1	ENT				
X SET	SHFT	I 8	→	F 5	ENT				
\$ STR	SHFT	I 8	→	D 3	ENT				
X SET	SHFT	I 8	→	H 7	ENT				
SHFT	I 8	R ORN	T MLR	ENT					

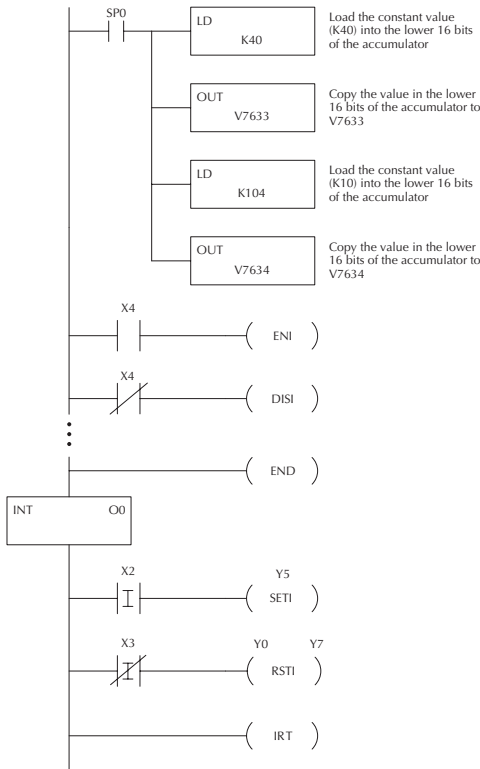
### Timed Interrupt Program Example

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 O 0. 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.

DirectSOFT 5

5



Handheld Programmer Keystrokes

\$ STR	→	B 1	ENT
SHFT	L ANDST	D 3	→
SHFT	K JMP	E 4	A 0
SHFT	ENT		
GX OUT	→	SHFT	V AND
H 7	G 6	D 3	D 3
SHFT	L ANDST	D 3	→
SHFT	K JMP	B 1	A 0
SHFT	ENT		
GX OUT	→	SHFT	V AND
H 7	G 6	D 3	E 4
\$ STR	→	E 4	ENT
SHFT	E 4	N TMR	I 8
SHFT	ENT		
SP STRN	→	E 4	ENT
SHFT	D 3	I 8	S RST
			I 8
			ENT

SHFT	E 4	N TMR	D 3	ENT
SHFT	I 8	N TMR	T MLR	→
SHFT	ENT			A 0
SHFT	ENT			
\$ STR	SHFT	I 8	→	C 2
SHFT	ENT			
X SET	SHFT	I 8	→	F 5
SHFT	ENT			
SP STRN	SHFT	I 8	→	D 3
SHFT	ENT			
X SET	SHFT	I 8	→	A 0
SHFT	ENT			
SHFT	I 8	R ORN	T MLR	→
SHFT	ENT			H 7
SHFT	ENT			

### Independent Timed Interrupt

Interrupt O1 is also available as an interrupt. This interrupt is independent of the HSIO features. Interrupt O1 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.

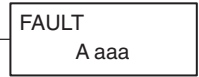
Input	Configuration Register	Function	Hex Code Required
-	V7647	High-Speed Timed Interrupt	xxxx (xxxx = timer setting) 5 - 9999 mS (BCD)

# Message Instructions

## Fault (FAULT)

DS5	Used
HPP	Used

The Fault instruction is used to display a message on the handheld programmer or in the *DirectSOFT* 5 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.

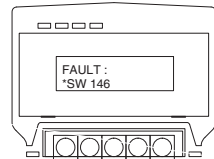
Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	See memory map
Constant ..... K	1-FFFF

Discrete Bit Flags	Description
SP50	On when the FAULT instruction is executed

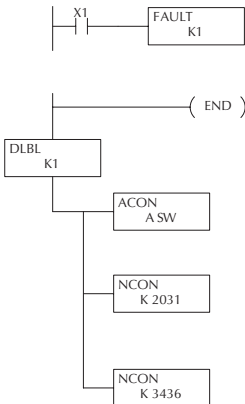
## Fault Example

DS5	Used
HPP	Used

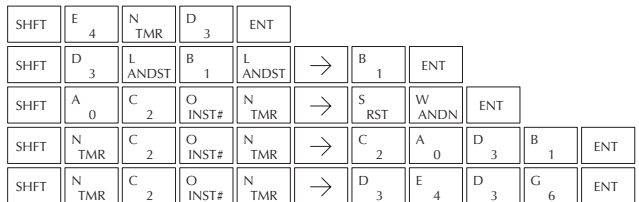
In the following example when X1 is on, the message SW 146 will display on the handheld programmer. The NCONs use the HEX ASCII equivalent of the text to be displayed. (The HEX ASCII for a blank is 20, a 1 is 31, 4 is 34 ...)



DirectSOFT 5



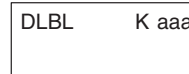
Handheld Programmer Keystrokes



### Data Label (DLBL)

DS5	Used
HPP	Used

The Data Label instruction marks the beginning of an ASCII/numeric data area. DLBLs are programmed after the End statement. A maximum of 64 DLBL instructions can be used in a program. Multiple NCONs and ACONs can be used in a DLBL area.



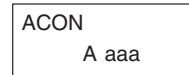
Operand Data Type	DL05 Range
..... A	aaa
Constant ..... K	1-FFFF

5

### ASCII Constant (ACON)

DS5	Used
HPP	Used

The ASCII Constant instruction is used with the DLBL instruction to store ASCII text for use with other instructions. Two ASCII characters can be stored in an ACON instruction. If only one character is stored in an ACON a leading space will be inserted.

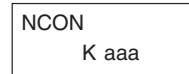


Operand Data Type	DL05 Range
..... A	aaa
Constant ..... A	0-9 A-Z

### Numerical Constant (NCON)

DS5	Used
HPP	Used

The Numerical Constant instruction is used with the DLBL instruction to store the HEX ASCII equivalent of numerical data for use with other instructions. Two digits can be stored in an NCON instruction.



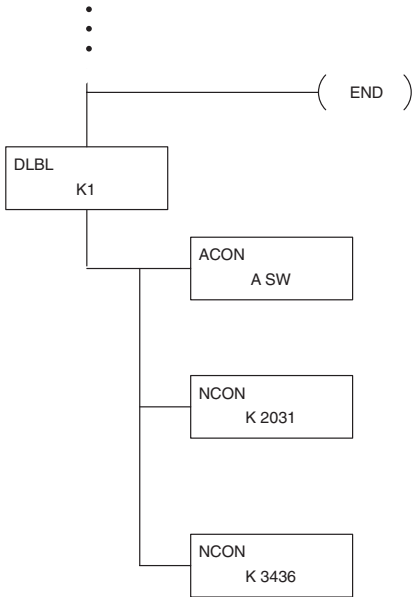
Operand Data Type	DL05 Range
..... K	aaa
Constant ..... K	0-FFFF



### 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.

DirectSOFT 5



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)

DS5	Used
HPP	N/A

The Print Message instruction prints the embedded text or text/data variable message to the specified, configured, communications port (Port 2 on the DL05 CPU).

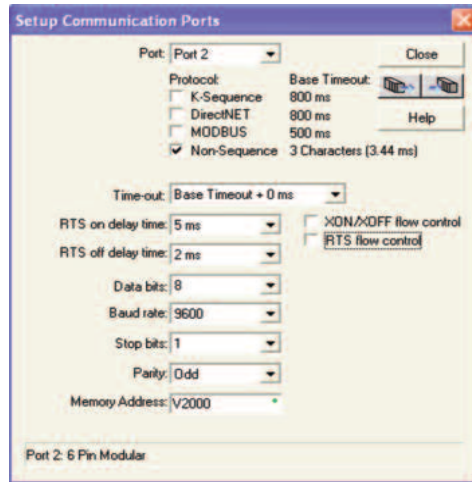
```
PRINT  A aaa
"Hello, this is a PLC message"
```

Operand Data Type	DL05 Range
.....A	aaa
Constant .....K	2

5

You may recall from the CPU specifications in Chapter 4 that the DL05'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 *DirectSOFT*, choose the PLC > Setup > 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 to be used by *DirectSOFT* to store the port setup information. You will need to reserve 66 contiguous words in V-memory for this purpose.



**NOTE:** See Chapter 4 for a detail of the non-sequence setup.



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 DL05.



**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:

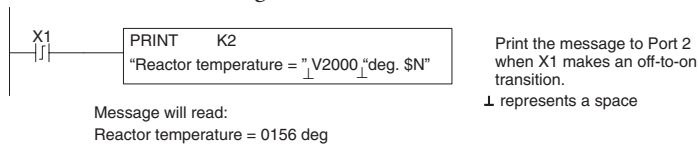
V2000            Print binary data in V2000 for decimal number

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 C00 in ON/OFF format**

**V2000.15 : BOOL**              **Prints the status of bit 15 in V2000 in TRUE/FALSE format**

The maximum numbers of characters you can print is 128. The number of characters for each element is listed in the table below:

Element Type	Maximum Characters
Text, 1 character	1
16 bit binary	6
32 bit binary	11
4 digit BCD	4
8 digit BCD	8
Floating point (real number)	12
Floating point (real with exponent)	12
V-memory/text	2
Bit (1/0 format)	1
Bit (TRUE/FALSE format)	5
Bit (ON/OFF format)	3

The handheld programmer's mnemonic is "PRINT," followed by the DEF field.

Special relay flags SP116 and SP117 indicate the status of the DL05 CPU ports (busy, or communications error). See the appendix on special relays for a description.



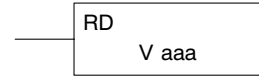
**NOTE:** You must use the appropriate special relay in conjunction with the PRINT command to ensure the ladder program does not try to PRINT to a port that is still busy from a previous PRINT or WX or RX instruction.

# Intelligent I/O Instructions

## Read from Intelligent Module (RD)

DS32	Used
HPP	Used

The Read from Intelligent Module instruction reads a block of data (1-128 bytes maximum) from an intelligent I/O module into the CPU's V-memory. It loads the function parameters into the first and second level of the accumulator stack and the accumulator by three additional instructions.



Listed below are the steps to program the Read from Intelligent module function.

Step 1: — Load the base number (0-3) into the first byte and the slot number (0-7) into the second byte of the second level of the accumulator stack.

Step 2: — Load the number of bytes to be transferred into the first level of the accumulator stack (maximum of 128 bytes).

Step 3: — Load the address from which the data will be read into the accumulator. This parameter must be a HEX value.

Step 4: — Insert the RD instruction which specifies the starting V-memory location (Vaaa) where the data will be read into.

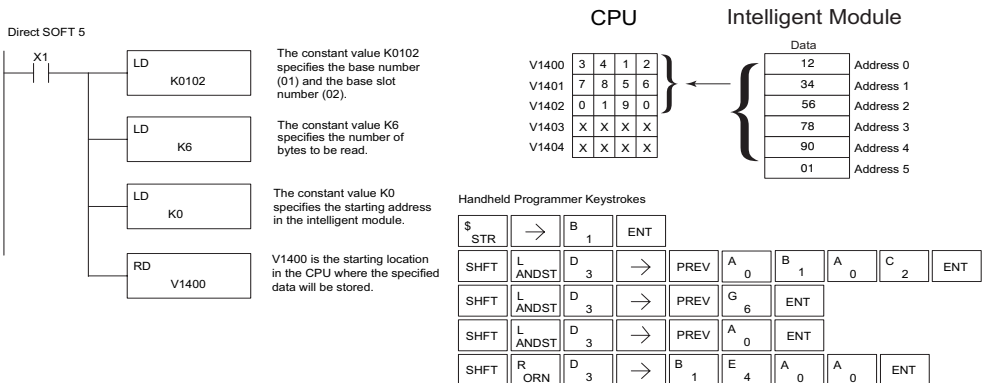
Helpful Hint: — Use the LDA instruction to convert an octal address to its HEX equivalent and load it into the accumulator when the HEX format is required.

Operand Data Type	DL06 Range
V-memory . . . . . V	aaa See memory map
Discrete Bit Flags	Description
SP54	On when RX, WX RD, WT instructions are executed with the wrong parameters.



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

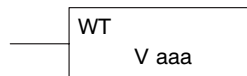
In the following example when X1 is ON, the RD instruction will read six bytes of data from a intelligent module in base 1, slot 2 starting at address 0 in the intelligent module and copy the information into V-memory locations V1400-V1402.



### Write to Intelligent Module (WT)

DS32	Used
HPP	Used

The Write to Intelligent Module instruction writes a block of data (1-128 bytes maximum) to an intelligent I/O module from a block of V-memory in the CPU. The function parameters are loaded into the first and second level of the accumulator stack and the accumulator by three additional instructions.



Listed below are the steps to program the Read from Intelligent module function.

Step 1: — Load the base number (0-3) into the first byte and the slot number (0-7) into the second byte of the second level of the accumulator stack.

Step 2: — Load the number of bytes to be transferred into the first level of the accumulator stack (maximum of 128 bytes).

Step 3: — Load the intelligent module address which will receive the data into the accumulator. This parameter must be a HEX value.

Step 4: — Insert the WT instruction which specifies the starting V-memory location (Vaaa) where the data will be written from in the CPU.

Helpful Hint: — Use the LDA instruction to convert an octal address to its HEX equivalent and load it into the accumulator when the HEX format is required.

Operand Data Type	DL06 Range
	<b>aaa</b>
V-memory ..... V	See memory map

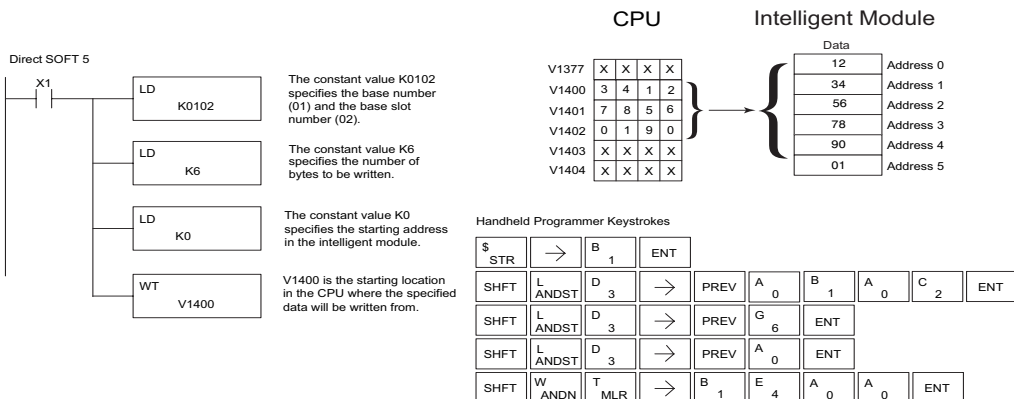
  

Discrete Bit Flags	Description
SP54	On when RX, WX RD, WT instructions are executed with the wrong parameters.



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

In the following example, when X1 is on, the WT instruction will write six bytes of data to an intelligent module in base 1, slot 2 starting at address 0 in the intelligent module and copy

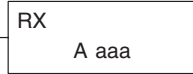


# Network Instructions

## Read from Network (RX)

DS5	Used
HPP	Used

The Read from Network instruction causes 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 accumulator and the first and second level of the stack. Listed below are the program steps necessary to execute the Read from Network function.



5

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, 2 - 128 bytes are allowed, (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 read from the slave will be put. This parameter requires a HEX value.

Step 4: — Insert the RX instruction which specifies the starting V-memory location (Aaaa) where the data will be read from in the slave.

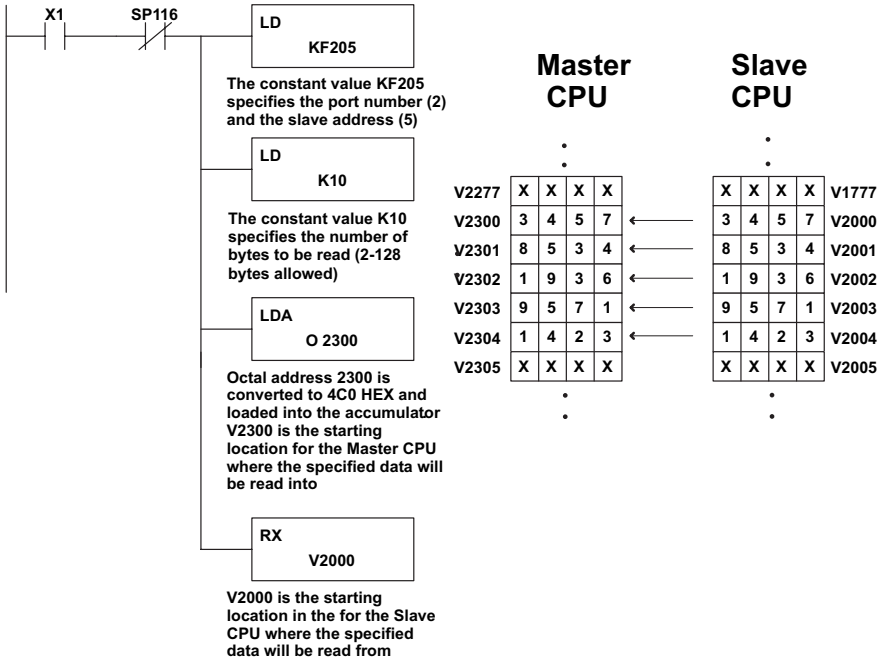
Helpful Hint: For parameters that require HEX values, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type	DL05 Range
<b>A</b>	<b>aaa</b>
V-memory . . . . . V	All (See page 3–28)
Pointer . . . . . P	All V-memory. (See page 3–28)
Inputs . . . . . X	0–377
Outputs . . . . . Y	0–377
Control Relays . . . . . C	0–777
Stage . . . . . S	0–377
Timer . . . . . T	0–177
Counter . . . . . CT	0–177
Special Relay . . . . . SP	0–777
Program Memory . . . . . \$	0–2047 (2K program mem.)

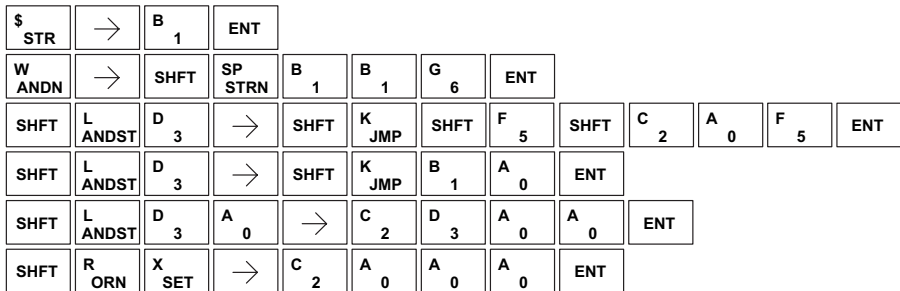


In the following example, when X1 is on and the port busy relay SP116 (see special relays) is not on, the RX instruction will access port 2 operating as a master. Ten consecutive bytes of data (V2000 – V2004) will be read from a CPU at station address 5 and copied into V-memory locations V2300–V2304 in the CPU with the master port.

DirectSOFT 5



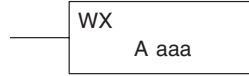
HandheldProgrammer Keystrokes



### Write to Network (WX)

DS5	Used
HPP	Used

The Write to Network instruction is used to write a block of data from the master device to a slave device on the same network. The function parameters are loaded into the 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, 2-128 bytes are allowed, (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.

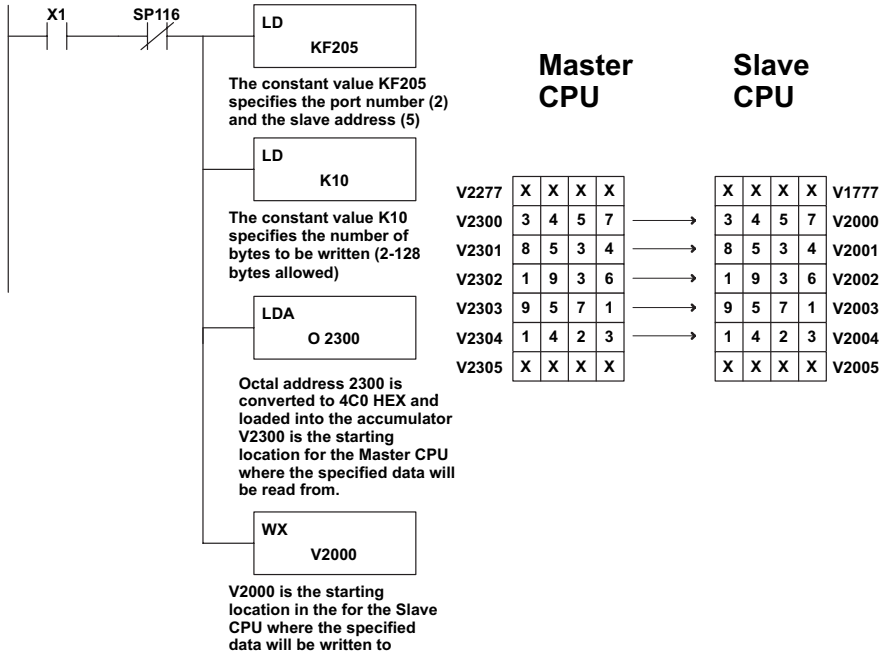
Step 4: — Insert the WX instruction which specifies the starting V-memory location (Aaaa) where the data will be written to in the slave.

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.

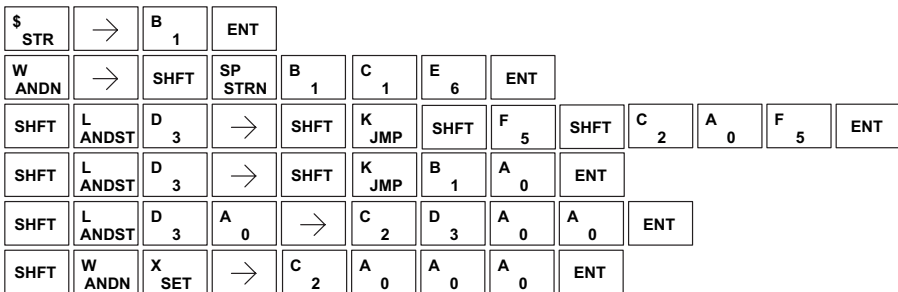
Operand Data Type	DL05 Range
..... A	aaa
V-memory ..... V	All (See page 3–28)
Pointer ..... P	All V-memory (See page 3–28)
Inputs ..... X	0–377
Outputs ..... Y	0–377
Control Relays ..... C	0–777
Stage ..... S	0–377
Timer ..... T	0–177
Counter ..... CT	0–177
Special Relay ..... SP	0–777
Program Memory ..... \$	0–2048 (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.

DirectSOFT 5



Handheld Programmer Keystrokes



## Intelligent Box (IBox) Instructions

The Intelligent Box Instructions (commonly referred to as IBox Instructions) listed in this section are additional, and much different looking, instructions made available with the release of **DirectSOFT 5**. The DL05 PLC requires firmware version v5.10 or later to use the new *DirectSOFT 5* features. For more information on *DirectSOFT 5*, please visit our website at: [www.automationdirect.com](http://www.automationdirect.com).

Analog Helper IBoxes		
Instruction	Ibox #	Page
Analog Input / Output Combo Module Pointer Setup (ANLGCMB)	IB-462	5-126
Analog Input Module Pointer Setup (ANLGIN)	IB-460	5-128
Analog Output Module Pointer Setup (ANLGOUT)	IB-461	5-130
Analog Scale 12 Bit BCD to BCD (ANSCL)	IB-423	5-132
Analog Scale 12 Bit Binary to Binary (ANSCLB)	IB-403	5-134
Filter Over Time - BCD (FILTER)	IB-422	5-136
Filter Over Time - Binary (FILTERB)	IB-402	5-138
Hi/Low Alarm - BCD (HILOAL)	IB-421	5-140
Hi/Low Alarm - Binary (HILOALB)	IB-401	5-142

Discrete Helper IBoxes		
Instruction	Ibox #	Page
Off Delay Timer (OFFDTMR)	IB-302	5-144
On Delay Timer (ONDTMR)	IB-301	5-146
One Shot (ONESHOT)	IB-303	5-148
Push On / Push Off Circuit (PONOFF)	IB-300	5-149

Memory IBoxes		
Instruction	Ibox #	Page
Move Single Word (MOVEW)	IB-200	5-150
Move Double Word (MOVED)	IB-201	5-151

Math IBoxes		
Instruction	Ibox #	Page
Math - BCD (MATHBCD)	IB-521	5-152
Math - Binary (MATHBIN)	IB-501	5-154
Square BCD (SQUARE)	IB-523	5-156
Square Binary (SQUAREB)	IB-503	5-157
Sum BCD Numbers (SUMBCD)	IB-522	5-158
Sum Binary Numbers (SUMBIN)	IB-502	5-159

<b>Communication IBoxes</b>		
<b>Instruction</b>	<b>Ibox #</b>	<b>Page</b>
ECOM100 Configuration (ECOM100)	IB-710	5-160
ECOM100 Disable DHCP (ECDHCPD)	IB-736	5-162
ECOM100 Enable DHCP (ECDHCPE)	IB-735	5-164
ECOM100 Query DHCP Setting (ECDHCPQ)	IB-734	5-166
ECOM100 Send E-mail (ECEMAIL)	IB-711	5-168
ECOM100 Restore Default E-mail Setup (ECEMRDS)	IB-713	5-171
ECOM100 E-mail Setup (ECEMSUP)	IB-712	5-174
ECOM100 IP Setup (ECIPSUP)	IB-717	5-178
ECOM100 Read Description (ECRDDDES)	IB-726	5-180
ECOM100 Read Gateway Address (ECRDGWA)	IB-730	5-182
ECOM100 Read IP Address (ECRDIP)	IB-722	5-184
ECOM100 Read Module ID (ECRDMID)	IB-720	5-186
ECOM100 Read Module Name (ECRDNAM)	IB-724	5-188
ECOM100 Read Subnet Mask (ECRDSNM)	IB-732	5-190
ECOM100 Write Description (ECWRDES)	IB-727	5-192
ECOM100 Write Gateway Address (ECWRGWA)	IB-731	5-194
ECOM100 Write IP Address (ECWRIP)	IB-723	5-196
ECOM100 Write Module ID (ECWRMID)	IB-721	5-198
ECOM100 Write Name (ECWRNAM)	IB-725	5-200
ECOM100 Write Subnet Mask (ECWRSNM)	IB-733	5-202
ECOM100 RX Network Read (ECRX)	IB-740	5-204
ECOM100 WX Network Write(ECWX)	IB-741	5-207
NETCFG Network Configuration (NETCFG)	IB-700	5-210
Network RX Read (NETRX)	IB-701	5-212
Network WX Write (NETWX)	IB-702	5-215

<b>Counter I/O IBoxes (Work with H0-CTRIO and H0-CTRIO2)</b>		
<b>Instruction</b>	<b>Ibox #</b>	<b>Page</b>
CTRIO Configuration (CTRIO)	IB-1000	5-218
CTRIO Add Entry to End of Preset Table (CTRADPT)	IB-1005	5-220
CTRIO Clear Preset Table (CTRCLRT)	IB-1007	5-223
CTRIO Edit Preset Table Entry (CTREDPT)	IB-1003	5-226
CTRIO Edit Preset Table Entry and Reload (CTREDRL)	IB-1002	5-230
CTRIO Initialize Preset Table (CTRINPT)	IB-1004	5-234
CTRIO Initialize Preset Table (CTRINTR)	IB-1010	5-238
CTRIO Load Profile (CTRLDPR)	IB-1001	5-242
CTRIO Read Error (CTRRDER)	IB-1014	5-244
CTRIO Run to Limit Mode (CTRRTLML)	IB-1011	5-246
CTRIO Run to Position Mode (CTRRTPM)	IB-1012	5-249
CTRIO Velocity Mode (CTRVELO)	IB-1013	5-251
CTRIO Write File to ROM (CTRWFTR)	IB-1006	5-254

## Analog Input/Output Combo Module Pointer Setup (ANLGCMB) (IB-462)

DS5	Used
HPP	N/A

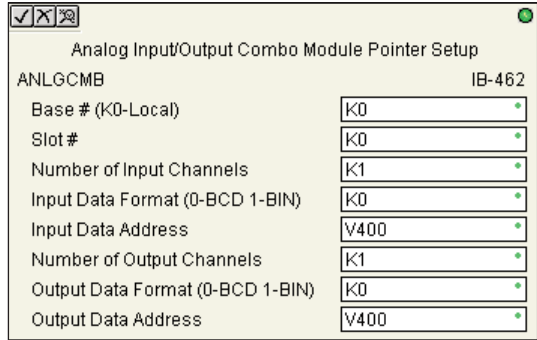
The Analog Input/Output Combo Module Pointer Setup instruction generates the logic to configure the pointer method for an analog input/output combination module on the first PLC scan following a Program to Run transition.

The ANLGCMB IBox instruction determines the data format and Pointer addresses based on the CPU type, the Base# and the module Slot#.

The Input Data Address is the starting location in user V-memory where the analog input data values will be stored, one location for each input channel enabled.

The Output Data Address is the starting location in user V-memory where the analog output data values will be placed by ladder code or external device, one location for each output channel enabled.

Since the IBox logic only executes on the first scan, the instruction cannot have any input logic.



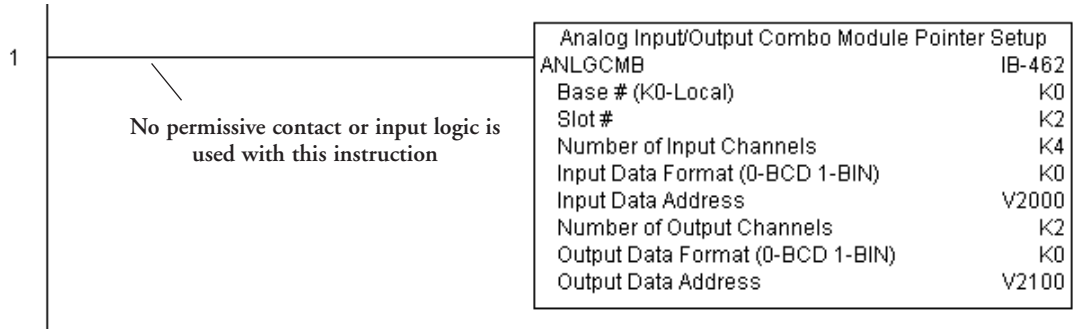
### ANLGCMB Parameters

- Base # (K0-Local): must be 0 for DL05 PLC
- Slot #: specifies the single PLC option slot that is occupied by the module
- Number of Input Channels: specifies the number of analog input channels to scan
- Input Data Format (0-BCD 1-BIN): specifies the analog input data format (BCD or Binary) - the binary format may be used for displaying data on some OI panels
- Input Data Address: specifies the starting V-memory location that will be used to store the analog input data
- Number of Output Channels: specifies the number of analog output channels that will be used
- Output Data Format (0-BCD 1-BIN): specifies the format of the analog output data (BCD or Binary)
- Output Data Address: specifies the starting V-memory location that will be used to source the analog output data

Parameter	DL05 Range
Base # (K0-Local) . . . . . K	K0 (local base only)
Slot # . . . . . K	K1
Number of Input Channels . . . . . K	K1-8
Input Data Format (0-BCD 1-BIN) . . . . . K	BCD: K0; Binary: K1
Input Data Address . . . . . V	See DL05 V-memory map - Data Words
Number of Output Channels . . . . . K	K1-8
Output Data Format (0-BCD 1-BIN) . . . . . K	BCD: K0; Binary: K1
Output Data Address . . . . . V	See DL05 V-memory map - Data Words

### ANLGCMB Example

In the following example, the ANLGCMB instruction is used to setup the pointer method for an analog I/O combination module that is installed in option slot 2. Four input channels are enabled and the analog data will be written to V2000 - V2003 in BCD format. Two output channels are enabled and the analog values will be read from V2100 - V2101 in BCD format.



### Analog Input Module Pointer Setup (ANLGIN) (IB-460)

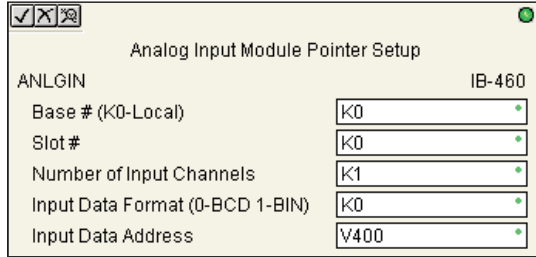
DS5	Used
HPP	N/A

Analog Input Module Pointer Setup generates the logic to configure the pointer method for one analog input module on the first PLC scan following a Program to Run transition.

This IBox determines the data format and Pointer addresses based on the CPU type, the Base#, and the Slot#.

The Input Data Address is the starting location in user V-memory where the analog input data values will be stored, one location for each input channel enabled.

Since this logic only executes on the first scan, this IBox cannot have any input logic.



#### ANLGIN Parameters

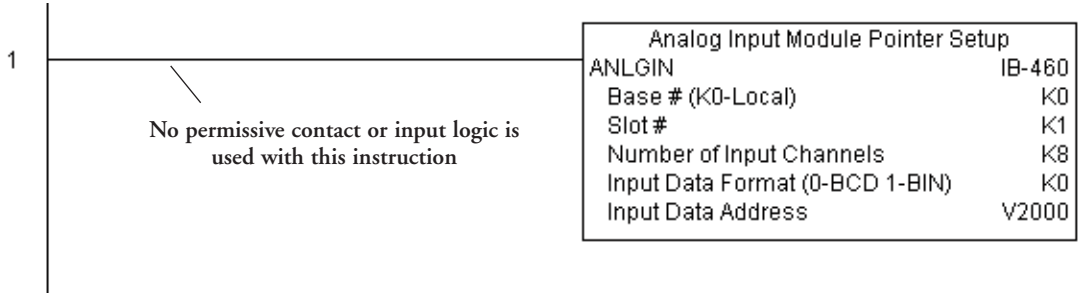
- Base # (K0-Local): must be 0 for DL05 PLC
- Slot #: specifies the single PLC option slot that is occupied by the module
- Number of Input Channels: specifies the number of input channels to scan
- Input Data Format (0-BCD 1-BIN): specifies the analog input data format (BCD or Binary) - the binary format may be used for displaying data on some OI panels
- Input Data Address: specifies the starting V-memory location that will be used to store the analog input data

Parameter	DL05 Range
Base # (K0-Local) . . . . . K	K0 (local base only)
Slot # . . . . . K	K1
Number of Input Channels . . . . . K	K1-8
Input Data Format (0-BCD 1-BIN) . . . . . K	BCD: K0; Binary: K1
Input Data Address . . . . . V	See DL05 V-memory map - Data Words



### ANLGIN Example

In the following example, the ANLGIN instruction is used to setup the pointer method for an analog input module that is installed in option slot 1. Eight input channels are enabled and the analog data will be written to V2000 - V2007 in BCD format.



### Analog Output Module Pointer Setup (ANLGOUT) (IB-461)

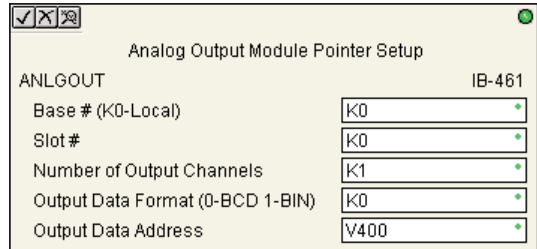
DS5	Used
HPP	N/A

Analog Output Module Pointer Setup generates the logic to configure the pointer method for one analog output module on the first PLC scan following a Program to Run transition.

This IBox determines the data format and Pointer addresses based on the CPU type, the Base#, and the Slot#.

The Output Data Address is the starting location in user V-memory where the analog output data values will be placed by ladder code or external device, one location for each output channel enabled.

Since this logic only executes on the first scan, this IBox cannot have any input logic.



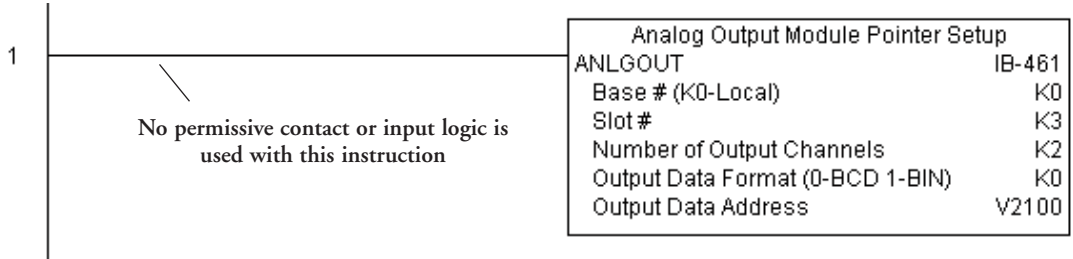
#### ANLGOUT Parameters

- Base # (K0-Local): must be 0 for DL05 PLC
- Slot #: specifies the single PLC option slot that is occupied by the module
- Number of Output Channels: specifies the number of analog output channels that will be used
- Output Data Format (0-BCD 1-BIN): specifies the format of the analog output data (BCD or Binary)
- Output Data Address: specifies the starting V-memory location that will be used to source the analog output data

Parameter	DL05 Range
Base # (K0-Local) . . . . . K	K0 (local base only)
Slot # . . . . . K	K1
Number of Output Channels . . . . . K	K1-8
Output Data Format (0-BCD 1-BIN) . . . . . K	BCD: K0; Binary: K1
Output Data Address . . . . . V	See DL05 V-memory map - Data Words

### ANLGOUT Example

In the following example, the ANLGOUT instruction is used to setup the pointer method for an analog output module that is installed in option slot 3. Two output channels are enabled and the analog data will be read from V2100 - V2101 in BCD format.

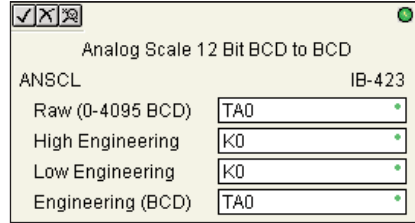


**Analog Scale 12 Bit BCD to BCD (ANSCL) (IB-423)**

DS5	Used
HPP	N/A

Analog Scale 12 Bit BCD to BCD scales a 12 bit BCD analog value (0-4095 BCD) into BCD engineering units. You specify the engineering unit high value (when raw is 4095), and the engineering low value (when raw is 0), and the output V memory address you want the to place the scaled engineering unit value. The engineering units are generated as BCD and can be the full range of 0 to 9999 (see ANSCLB - Analog Scale 12 Bit Binary to Binary if your raw units are in Binary format).

Note that this IBox only works with unipolar unsigned raw values. It does NOT work with bipolar or sign plus magnitude raw values.



5

**ANSCL Parameters**

- Raw (0-4095 BCD): specifies the V-memory location of the unipolar unsigned raw 0-4095 unscaled value
- High Engineering: specifies the high engineering value when the raw input is 4095
- Low Engineering: specifies the low engineering value when the raw input is 0
- Engineering (BCD): specifies the V-memory location where the scaled engineering BCD value will be placed

Parameter	DL05 Range
Raw (0-4095 BCD) . . . . . V,P	See DL05 V-memory map - Data Words
High Engineering . . . . . K	K0-9999
Low Engineering . . . . . K	K0-9999
Engineering (BCD) . . . . . V,P	See DL05 V-memory map - Data Words

### ANSCL Example

In the following example, the ANSCL instruction is used to scale a raw value (0-4095 BCD) that is in V2000. The engineering scaling range is set 0-100 (low engineering value - high engineering value). The scaled value will be placed in V2100 in BCD format.

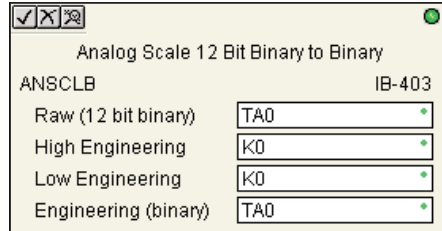


**Analog Scale 12 Bit Binary to Binary (ANSCLB) (IB-403)**

DS5	Used
HPP	N/A

Analog Scale 12 Bit Binary to Binary scales a 12 bit binary analog value (0-4095 decimal) into binary (decimal) engineering units. You specify the engineering unit high value (when raw is 4095), and the engineering low value (when raw is 0), and the output V-memory address you want to place the scaled engineering unit value. The engineering units are generated as binary and can be the full range of 0 to 65535 (see ANSCL - Analog Scale 12 Bit BCD to BCD if your raw units are in BCD format).

Note that this IBox only works with unipolar unsigned raw values. It does NOT work with bipolar, sign plus magnitude, or signed 2's complement raw values.



**ANSCLB Parameters**

- Raw (12 bit binary): specifies the V-memory location of the unipolar unsigned raw decimal unscaled value (12 bit binary = 0-4095 decimal)
- High Engineering: specifies the high engineering value when the raw input is 4095 decimal
- Low Engineering: specifies the low engineering value when the raw input is 0 decimal
- Engineering (binary): specifies the V-memory location where the scaled engineering decimal value will be placed

Parameter	DL05 Range
Raw (12 bit binary) . . . . . V,P	See DL05 V-memory map - Data Words
High Engineering . . . . . K	K0-65535
Low Engineering . . . . . K	K0-65535
Engineering (binary) . . . . . V,P	See DL05 V-memory map - Data Words

### ANSCLB Example

In the following example, the ANSCLB instruction is used to scale a raw value (0-4095 binary) that is in V2000. The engineering scaling range is set 0-1000 (low engineering value - high engineering value). The scaled value will be placed in V2100 in binary format.



### Filter Over Time - BCD (FILTER) (IB-422)

DS5	Used
HPP	N/A

Filter Over Time BCD will perform a first-order filter on the Raw Data on a defined time interval. The equation is:

$New = Old + [(Raw - Old) / FDC]$  where,

New: New Filtered Value

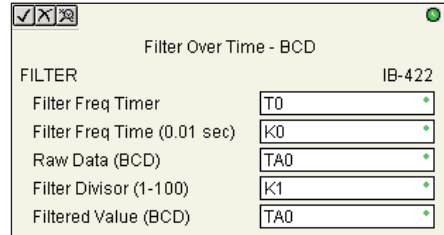
Old: Old Filtered Value

FDC: Filter Divisor Constant

Raw: Raw Data

The Filter Divisor Constant is an integer in the range K1 to K100, such that if it equaled K1 then no filtering would be done.

The rate at which the calculation is performed is specified by time in hundredths of a second (0.01 seconds) as the Filter Freq Time parameter. Note that this Timer instruction is embedded in the IBox and must NOT be used anywhere else in your program. Power flow controls whether the calculation is enabled. If it is disabled, the Filter Value is not updated. On the first scan from Program to Run mode, the Filter Value is initialized to 0 to give the calculation a consistent starting point.



#### FILTER Parameters

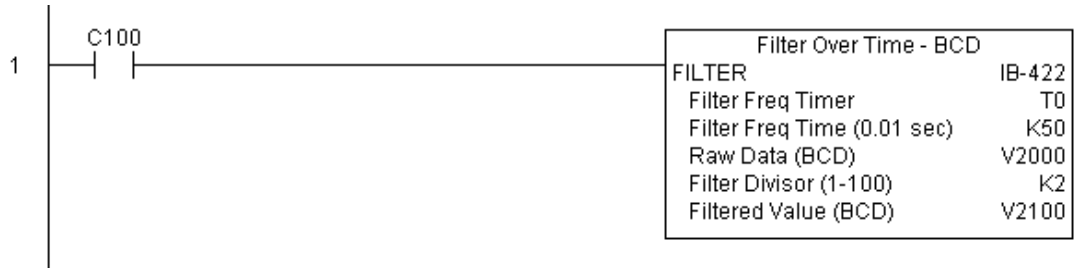
- Filter Frequency Timer: specifies the Timer (T) number which is used by the Filter instruction
- Filter Frequency Time (0.01sec): specifies the rate at which the calculation is performed
- Raw Data (BCD): specifies the V-memory location of the raw unfiltered BCD value
- Filter Divisor (1-100): this constant used to control the filtering effect. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering.
- Filtered Value (BCD): specifies the V-memory location where the filtered BCD value will be placed

Parameter	DL05 Range
Filter Frequency Timer . . . . . T	T0-177
Filter Frequency Time (0.01 sec) . . . . . K	K0-9999
Raw Data (BCD) . . . . . V	See DL05 V-memory map - Data Words
Filter Divisor (1-100) . . . . . K	K1-100
Filtered Value (BCD) . . . . . V	See DL05 V-memory map - Data Words



### FILTER Example

In the following example, the Filter instruction is used to filter a BCD value that is in V2000. Timer(T0) is set to 0.5 sec, the rate at which the filter calculation will be performed. The filter constant is set to 2. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering. The filtered value will be placed in V2100.



### Filter Over Time - Binary (FILTERB) (IB-402)

Filter Over Time in Binary (decimal) will perform a first-order filter on the Raw Data on a defined time interval. The equation is:

DS5	Used
HPP	N/A

New = Old + [(Raw - Old) / FDC] where

New: New Filtered Value

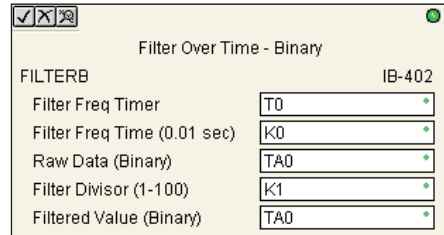
Old: Old Filtered Value

FDC: Filter Divisor Constant

Raw: Raw Data

The Filter Divisor Constant is an integer in the range K1 to K100, such that if it equaled K1 then no filtering would be done.

The rate at which the calculation is performed is specified by time in hundredths of a second (0.01 seconds) as the Filter Freq Time parameter. Note that this Timer instruction is embedded in the IBox and must NOT be used anywhere else in your program. Power flow controls whether the calculation is enabled. If it is disabled, the Filter Value is not updated. On the first scan from Program to Run mode, the Filter Value is initialized to 0 to give the calculation a consistent starting point.



#### FILTERB Parameters

- Filter Frequency Timer: specifies the Timer (T) number which is used by the Filter instruction
- Filter Frequency Time (0.01sec): specifies the rate at which the calculation is performed
- Raw Data (Binary): specifies the V-memory location of the raw unfiltered binary (decimal) value
- Filter Divisor (1-100): this constant used to control the filtering effect. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering.
- Filtered Value (Binary): specifies the V-memory location where the filtered binary (decimal) value will be placed

Parameter	DL05 Range
Filter Frequency Timer . . . . . T	T0-177
Filter Frequency Time (0.01 sec) . . . . . K	K0-9999
Raw Data (Binary) . . . . . V	See DL05 V-memory map - Data Words
Filter Divisor (1-100) . . . . . K	K1-100
Filtered Value (Binary) . . . . . V	See DL05 V-memory map - Data Words

### FILTERB Example

In the following example, the FILTERB instruction is used to filter a binary value that is in V2000. Timer(T1) is set to 0.5 sec, the rate at which the filter calculation will be performed. The filter constant is set to 3. A larger value will increase the smoothing effect of the filter. A value of 1 results with no filtering. The filtered value will be placed in V2100.

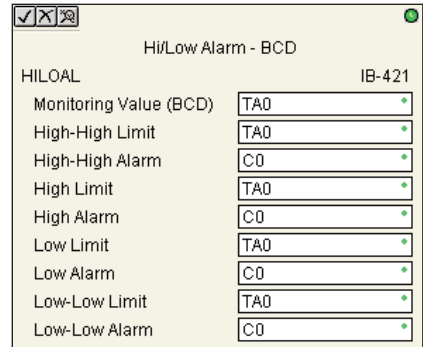


### Hi/Low Alarm - BCD (HILOAL) (IB-421)

DS5	Used
HPP	N/A

Hi/Low Alarm - BCD monitors a BCD value V-memory location and sets four possible alarm states, High-High, High, Low, and Low-Low whenever the IBox has power flow. You enter the alarm thresholds as constant K BCD values (K0-K9999) and/or BCD value V-memory locations.

You must ensure that threshold limits are valid, that is  $HH \geq H > L \geq LL$ . Note that when the High-High or Low-Low alarm condition is true, that the High and Low alarms will also be set, respectively. This means you may use the same threshold limit and same alarm bit for the High-High and the High alarms in case you only need one "High" alarm. Also note that the boundary conditions are inclusive. That is, if the Low boundary is K50, and the Low-Low boundary is K10, and if the Monitoring Value equals 10, then the Low Alarm AND the Low-Low alarm will both be ON. If there is no power flow to the IBox, then all alarm bits will be turned off regardless of the value of the Monitoring Value parameter.



#### HILOAL Parameters

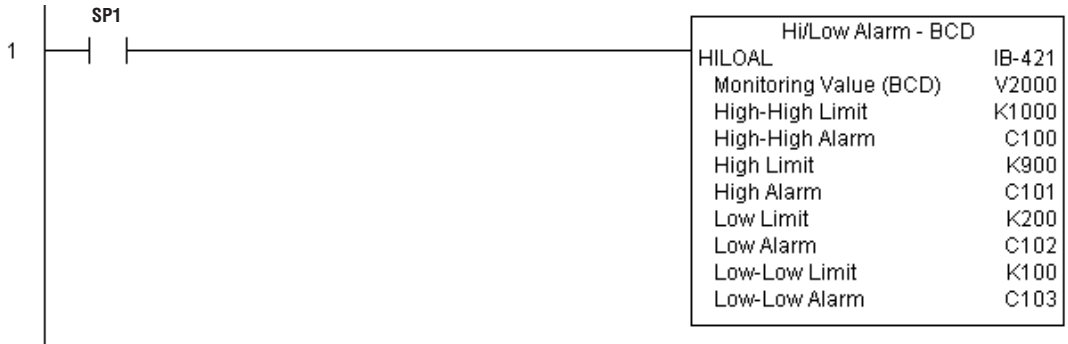
- Monitoring Value (BCD): specifies the V-memory location of the BCD value to be monitored
- High-High Limit: V-memory location or constant specifies the high-high alarm limit
- High-High Alarm: On when the high-high limit is reached
- High Limit: V-memory location or constant specifies the high alarm limit
- High Alarm: On when the high limit is reached
- Low Limit: V-memory location or constant specifies the low alarm limit
- Low Alarm: On when the low limit is reached
- Low-Low Limit: V-memory location or constant specifies the low-low alarm limit
- Low-Low Alarm: On when the low-low limit is reached

Parameter	DL05 Range
Monitoring Value (BCD) . . . . . V	See DL05 V-memory map - Data Words
High-High Limit . . . . . V, K	K0-9999; or see DL05 V-memory map - Data Words
High-High Alarm . . . . . X, Y, C, GX,GY, B	See DL05 V-memory map
High Limit . . . . . V, K	K0-9999; or see DL05 V-memory map - Data Words
High Alarm . . . . . X, Y, C, GX,GY, B	See DL05 V-memory map
Low Limit . . . . . V, K	K0-9999; or see DL05 V-memory map - Data Words
Low Alarm . . . . . X, Y, C, GX,GY,B	See DL05 V-memory map
Low-Low Limit . . . . . V, K	K0-9999; or see DL05 V-memory map - Data Words
Low-Low Alarm. . . . . X, Y, C, GX,GY, B	See DL05 V-memory map

### HILOAL Example

In the following example, the HILOAL instruction is used to monitor a BCD value that is in V2000. If the value in V2000 meets/exceeds the high limit of K900, C101 will turn on. If the value continues to increase to meet/exceed the high-high limit, C100 will turn on. Both bits would be on in this case. The high and high-high limits and alarms can be set to the same value if one “high” limit or alarm is desired to be used.

If the value in V2000 meets or falls below the low limit of K200, C102 will turn on. If the value continues to decrease to meet or fall below the low-low limit of K100, C103 will turn on. Both bits would be on in this case. The low and low-low limits and alarms can be set to the same value if one “low” limit or alarm is desired to be used.

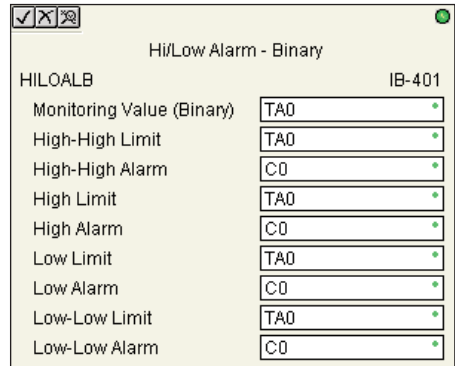


## Hi/Low Alarm - Binary (HILOALB) (IB-401)

DS5	Used
HPP	N/A

Hi/Low Alarm - Binary monitors a binary (decimal) V-memory location and sets four possible alarm states, High-High, High, Low, and Low-Low whenever the IBox has power flow. You enter the alarm thresholds as constant K decimal values (K0-K65535) and/or binary (decimal) V-memory locations.

You must ensure that threshold limits are valid, that is  $HH \geq H > L \geq LL$ . Note that when the High-High or Low-Low alarm condition is true, that the High and Low alarms will also be set, respectively. This means you may use the same threshold limit and same alarm bit for the High-High and the High alarms in case you only need one "High" alarm. Also note that the boundary conditions are inclusive. That is, if the Low boundary is K50, and the Low-Low boundary is K10, and if the Monitoring Value equals 10, then the Low Alarm AND the Low-Low alarm will both be ON. If there is no power flow to the IBox, then all alarm bits will be turned off regardless of the value of the Monitoring Value parameter.



### HILOALB Parameters

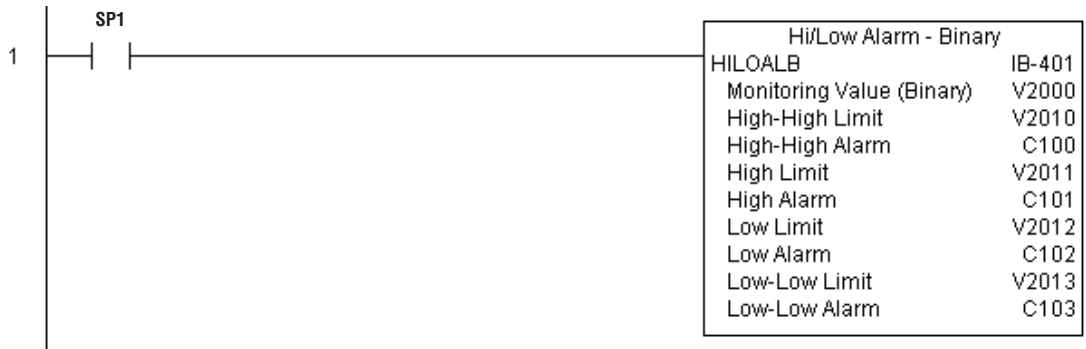
- Monitoring Value (Binary): specifies the V-memory location of the Binary value to be monitored
- High-High Limit: V-memory location or constant specifies the high-high alarm limit
- High-High Alarm: On when the high-high limit is reached
- High Limit: V-memory location or constant specifies the high alarm limit
- High Alarm: On when the high limit is reached
- Low Limit: V-memory location or constant specifies the low alarm limit
- Low Alarm: On when the low limit is reached
- Low-Low Limit: V-memory location or constant specifies the low-low alarm limit
- Low-Low Alarm: On when the low-low limit is reached

Parameter	DL05 Range
Monitoring Value (Binary) . . . . . V	See DL05 V-memory map - Data Words
High-High Limit . . . . . V, K	K0-65535; or see DL05 V-memory map - Data Words
High-High Alarm . . . . . X, Y, C, GX,GY, B	See DL05 V-memory map
High Limit . . . . . V, K	K0-65535; or see DL05 V-memory map - Data Words
High Alarm . . . . . X, Y, C, GX,GY, B	See DL05 V-memory map
Low Limit . . . . . V, K	K0-65535; or see DL05 V-memory map - Data Words
Low Alarm . . . . . X, Y, C, GX,GY,B	See DL05 V-memory map
Low-Low Limit . . . . . V, K	K0-65535; or see DL05 V-memory map - Data Words
Low-Low Alarm. . . . . X, Y, C, GX,GY, B	See DL05 V-memory map

### HILOALB Example

In the following example, the HILOALB instruction is used to monitor a binary value that is in V2000. If the value in V2000 meets/exceeds the high limit of the binary value in V2011, C101 will turn on. If the value continues to increase to meet/exceed the high-high limit value in V2010, C100 will turn on. Both bits would be on in this case. The high and high-high limits and alarms can be set to the same V-memory location/value if one “high” limit or alarm is desired to be used.

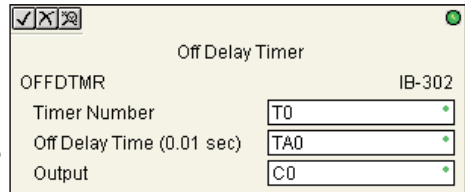
If the value in V2000 meets or falls below the low limit of the binary value in V2012, C102 will turn on. If the value continues to decrease to meet or fall below the low-low limit in V2013, C103 will turn on. Both bits would be on in this case. The low and low-low limits and alarms can be set to the same V-memory location/value if one “low” limit or alarm is desired to be used.



### Off Delay Timer (OFFDTMR) (IB-302)

DS5	Used
HPP	N/A

Off Delay Timer will delay the "turning off" of the Output parameter by the specified Off Delay Time (in hundredths of a second) based on the power flow into the IBox. Once the IBox receives power, the Output bit will turn on immediately. When the power flow to the IBox turns off, the Output bit WILL REMAIN ON for the specified amount of time (in hundredths of a second). Once the Off Delay Time has expired, the output will turn Off. If the power flow to the IBox comes back on BEFORE the Off Delay Time, then the timer is RESET and the Output will remain On - so you must continuously have NO power flow to the IBox for AT LEAST the specified Off Delay Time before the Output will turn Off.



This IBox utilizes a Timer resource (TMRF), which cannot be used anywhere else in your program.

#### OFFDTMR Parameters

- Timer Number: specifies the Timer(TMRF) number which is used by the OFFDTMR instruction
- Off Delay Time (0.01sec): specifies how long the Output will remain on once power flow to the Ibox is removed
- Output: specifies the output that will be delayed "turning off" by the Off Delay Time.

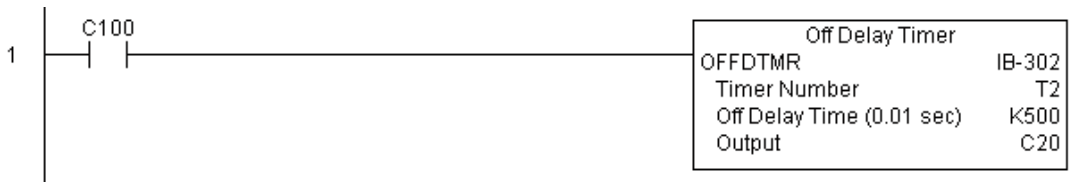
Parameter	DL05 Range
Timer Number . . . . . T	T0-177
Off Delay Time . . . . . K,V	K0-9999; See DL05 V-memory map - Data Words
Output . . . . . X, Y, C, GX,GY, B	See DL05 V-memory map



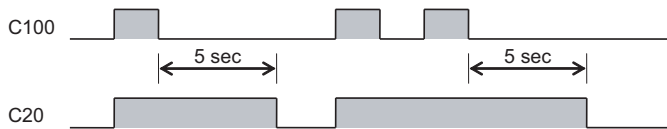
### OFFDTMR Example

In the following example, the OFFDTMR instruction is used to delay the “turning off” of output C20. Timer 2 (T2) is set to 5 seconds, the “off-delay” period.

When C100 turns on, C20 turns on and will remain on while C100 is on. When C100 turns off, C20 will remain for the specified Off Delay Time (5s), and then turn off.



Example timing diagram

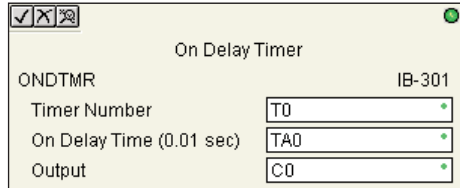


### On Delay Timer (ONDTMR) (IB-301)

On Delay Timer will delay the "turning on" of the Output parameter by the specified amount of time (in hundredths of a second) based on the power flow into the IBox. Once the IBox loses power, the Output is turned off immediately. If the power flow turns off BEFORE the On Delay Time, then the timer is RESET and the Output is never turned on, so you must have continuous power flow to the IBox for at least the specified On Delay Time before the Output turns On.

DS5	Used
HPP	N/A

This IBox utilizes a Timer resource (TMRF), which cannot be used anywhere else in your program.



#### ONDTMR Parameters

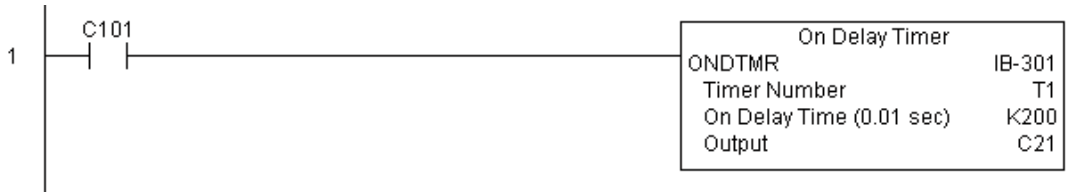
- Timer Number: specifies the Timer(TMRF) number which is used by the ONDTMR instruction
- On Delay Time (0.01sec): specifies how long the Output will remain on once power flow to the Ibox is removed
- Output: specifies the output that will be delayed "turning on" by the On Delay Time.

Parameter	DL05 Range
Timer Number ..... T	T0-177
On Delay Time ..... K,V	K0-9999; See DL05 V-memory map - Data Words
Output ..... X, Y, C, GX,GY, B	See DL05 V-memory map

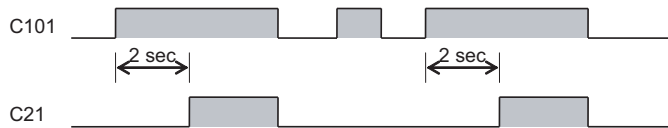
### ONDTMR Example

In the following example, the ONDTMR instruction is used to delay the “turning on” of output C21. Timer 1 (T1) is set to 2 seconds, the “on-delay” period.

When C101 turns on, C21 is delayed turning on by 2 seconds. When C101 turns off, C21 turns off immediately.



### Example timing diagram



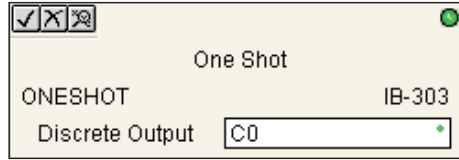
### One Shot (ONESHOT) (IB-303)

One Shot will turn on the given bit output parameter for one scan on an OFF to ON transition of the power flow into the IBox. This IBox is simply a different name for the PD Coil (Positive Differential).

DS5	Used
HPP	N/A

#### ONESHOT Parameters

- Discrete Output: specifies the output that will be on for one scan

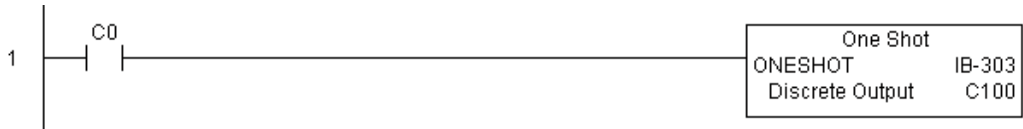


5

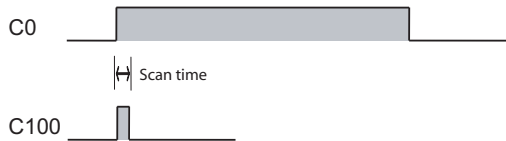
Parameter	DL05 Range
Discrete Output . . . . . X, Y, C	See DL05 V-memory map

### ONESHOT Example

In the following example, the ONESHOT instruction is used to turn C100 on for one PLC scan after C0 goes from an off to on transition. The input logic must produce an off to on transition to execute the One Shot instruction.



#### Example timing diagram



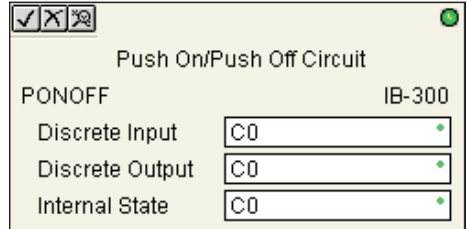
### Push On / Push Off Circuit (PONOFF) (IB-300)

DS5	Used
HPP	N/A

Push On/ Push Off Circuit toggles an output state whenever its input power flow transitions from off to on. Requires an extra bit parameter for scan-to-scan state information. This extra bit must NOT be used anywhere else in the program. This is also known as a “flip-flop circuit”.

#### PONOFF Parameters

- Discrete Input: specifies the input that will toggle the specified output
- Discrete Output: specifies the output that will be “turned on/off” or toggled
- Internal State: specifies a work bit that is used by the instruction



5

Parameter	DL05 Range
Discrete Input . . . . X,Y,C,S,T,CT,GX,GY,SP,B,PB	See DL05 V-memory map
Discrete Output . . . . . X,Y,C,GX,GY,B	See DL05 V-memory map
Internal State . . . . . X, Y, C	See DL05 V-memory map

### PONOFF Example

In the following example, the PONOFF instruction is used to control the on and off states of the output C20 with a single input C10. When C10 is pressed once, C20 turns on. When C10 is pressed again, C20 turns off. C100 is an internal bit used by the instruction.



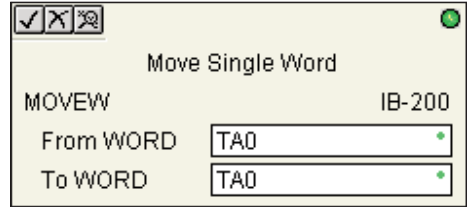
### Move Single Word (MOVEW) (IB-200)

Move Single Word moves (copies) a word to a memory location directly or indirectly via a pointer, either as a HEX constant, from a memory location, or indirectly through a pointer.

DS5	Used
HPP	N/A

#### MOVEW Parameters

- From WORD: specifies the word that will be moved to another location
- To WORD: specifies the location where the “From WORD” will be move to



Parameter	DL05 Range
From WORD .....	V,P,K
To WORD .....	V,P
	K0-FFFF; See DL05 V-memory map - Data Words
	See DL05 V-memory map - Data Words

### MOVEW Example

In the following example, the MOVEW instruction is used to move 16-bits of data from V2000 to V3000 when C100 turns on.



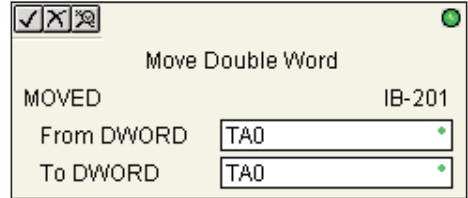
### Move Double Word (MOVED) (IB-201)

Move Double Word moves (copies) a double word to two consecutive memory locations directly or indirectly via a pointer, either as a double HEX constant, from a double memory location, or indirectly through a pointer to a double memory location.

DS5	Used
HPP	N/A

#### MOVED Parameters

- From DWORD: specifies the double word that will be moved to another location
- To DWORD: specifies the location where the “From DWORD” will be move to



5

Parameter	DL05 Range
From DWORD ..... V,P,K	K0-FFFFFFF; See DL05 V-memory map - Data Words
To DWORD ..... V,P	See DL05 V-memory map - Data Words

### MOVED Example

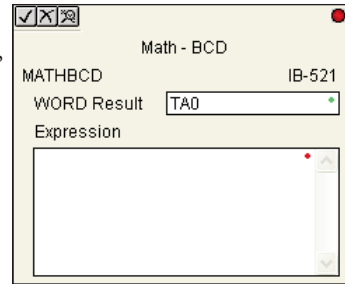
In the following example, the MOVED instruction is used to move 32-bits of data from V2000 and V2001 to V3000 and V3001 when C100 turns on.



**Math - BCD (MATHBCD) (IB-521)**

DS5	Used
HPP	N/A

Math - BCD Format lets you enter complex mathematical expressions like you would in Visual Basic, Excel, or C++ to do complex calculations, nesting parentheses up to 4 levels deep. In addition to + - \* /, you can do Modulo (% aka Remainder), Bit-wise And (&) Or (|) Xor (^), and some BCD functions - Convert to BCD (BCD), Convert to Binary (BIN), BCD Complement (BCDCPL), Convert from Gray Code (GRAY), Invert Bits (INV), and BCD/HEX to Seven Segment Display (SEG).



Example:  $((V2000 + V2001) / (V2003 - K100)) * GRAY(V3000 \& K001F)$

Every V-memory reference MUST be to a single word BCD formatted value. Intermediate results can go up to 32 bit values, but as long as the final result fits in a 16 bit BCD word, the calculation is valid. Typical example of this is scaling using multiply then divide,  $(V2000 * K1000) / K4095$ . The multiply term most likely will exceed 9999 but fits within 32 bits. The divide operation will divide 4095 into the 32-bit accumulator, yielding a result that will always fit in 16 bits.

You can reference binary V-memory values by using the BCD conversion function on a V-memory location but NOT an expression. That is  $BCD(V2000)$  is okay and will convert V2000 from Binary to BCD, but  $BCD(V2000 + V3000)$  will add V2000 as BCD, to V3000 as BCD, then interpret the result as Binary and convert it to BCD - NOT GOOD.

Also, the final result is a 16 bit BCD number and so you could do BIN around the entire operation to store the result as Binary.

**MATHBCD Parameters**

- WORD Result: specifies the location where the BCD result of the mathematical expression will be placed (result must fit into 16 bit single V-memory location)
- Expression: specifies the mathematical expression to be executed and the result is stored in specified WORD Result. Each V-memory location used in the expression must be in BCD format.

Parameter	DL05 Range
WORD Result . . . . . V	See DL05 V-memory map - Data Words
Expression . . . . .	Text



### MATHBCD Example

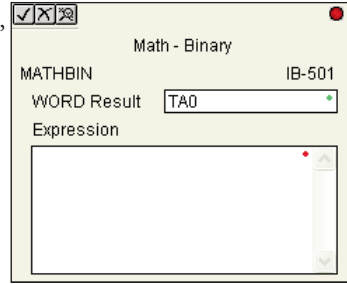
In the following example, the MATHBCD instruction is used to calculate the math expression which multiplies the BCD value in V1200 by 1000 then divides by 4095 and loads the resulting value in V2000.



### Math - Binary (MATHBIN) (IB-501)

DS5	Used
HPP	N/A

Math - Binary Format lets you enter complex mathematical expressions like you would in Visual Basic, Excel, or C++ to do complex calculations, nesting parentheses up to 4 levels deep. In addition to + - \* /, you can do Modulo (% aka Remainder), Shift Right (>>) and Shift Left (<<), Bit-wise And (&) Or (|) Xor (^), and some binary functions - Convert to BCD (BCD), Convert to Binary (BIN), Decode Bits (DECO), Encode Bits (ENCO), Invert Bits (INV), HEX to Seven Segment Display (SEG), and Sum Bits (SUM).



Example:  $((V2000 + V2001) / (V2003 - K10)) * SUM(V3000 \& K001F)$

Every V-memory reference MUST be to a single word binary formatted value. Intermediate results can go up to 32 bit values, but as long as the final result fits in a 16 bit binary word, the calculation is valid. Typical example of this is scaling using multiply then divide,  $(V2000 * K1000) / K4095$ . The multiply term most likely will exceed 65535 but fits within 32 bits. The divide operation will divide 4095 into the 32-bit accumulator, yielding a result that will always fit in 16 bits.

You can reference BCD V-memory values by using the BIN conversion function on a V-memory location but NOT an expression. That is,  $BIN(V2000)$  is okay and will convert V2000 from BCD to Binary, but  $BIN(V2000 + V3000)$  will add V2000 as Binary, to V3000 as Binary, then interpret the result as BCD and convert it to Binary - NOT GOOD.

Also, the final result is a 16 bit binary number and so you could do BCD around the entire operation to store the result as BCD.

#### MATHBIN Parameters

- WORD Result: specifies the location where the binary result of the mathematical expression will be placed (result must fit into 16 bit single V-memory location)
- Expression: specifies the mathematical expression to be executed and the result is stored in specified WORD Result. Each V-memory location used in the expression must be in binary format.

Parameter	DL05 Range
WORD Result .....	See DL05 V-memory map - Data Words
Expression .....	Text

### MATHBIN Example

In the following example, the MATHBIN instruction is used to calculate the math expression which multiplies the Binary value in V1200 by 1000 then divides by 4095 and loads the resulting value in V2000.



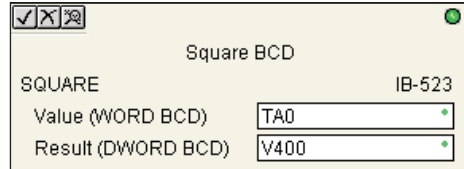
### Square BCD (SQUARE) (IB-523)

Square BCD squares the given 4-digit WORD BCD number and writes it in as an 8-digit DWORD BCD result.

DS5	Used
HPP	N/A

#### SQUARE Parameters

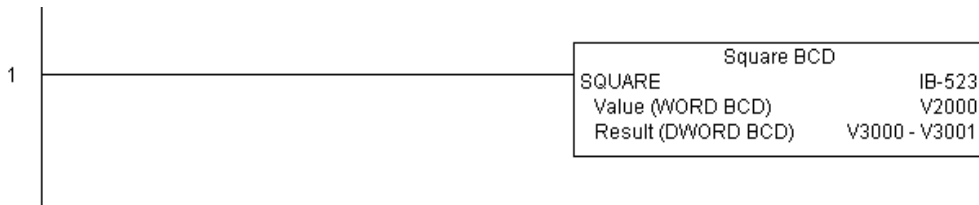
- Value (WORD BCD): specifies the BCD Word or constant that will be squared
- Result (DWORD BCD): specifies the location where the squared DWORD BCD value will be placed



Parameter	DL05 Range
Value (WORD BCD) . . . . . V,P,K	K0-9999 ; See DL05 V-memory map - Data Words
Result (DWORD BCD) . . . . . V	See DL05 V-memory map - Data Words

### SQUARE Example

In the following example, the SQUARE instruction is used to square the 4-digit BCD value in V2000 and store the 8-digit double word BCD result in V3000 and V3001



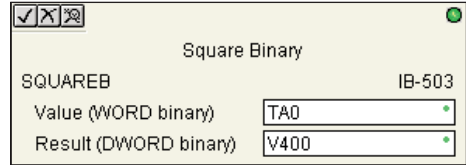
### Square Binary (SQUAREB) (IB-503)

Square Binary squares the given 16-bit WORD Binary number and writes it as a 32-bit DWORD Binary result.

DS5	Used
HPP	N/A

#### SQUAREB Parameters

- Value (WORD Binary): specifies the binary Word or constant that will be squared
- Result (DWORD Binary): specifies the location where the squared DWORD binary value will be placed

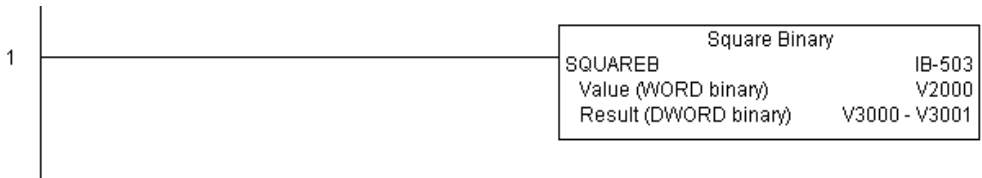


5

Parameter	DL05 Range
Value (WORD Binary) . . . . . V,P,K	K0-65535; See DL05 V-memory map - Data Words
Result (DWORD Binary) . . . . . V	See DL05 V-memory map - Data Words

### SQUAREB Example

In the following example, the SQUAREB instruction is used to square the single word Binary value in V2000 and store the 8-digit double word Binary result in V3000 and V3001.



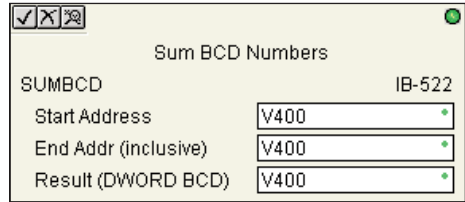
### Sum BCD Numbers (SUMBCD) (IB-522)

DS5	Used
HPP	N/A

Sum BCD Numbers sums up a list of consecutive 4-digit WORD BCD numbers into an 8-digit DWORD BCD result.

You specify the group's starting and ending V-memory addresses (inclusive). When enabled, this instruction will add up all the numbers in the group (so you may want to place a differential contact driving the enable).

SUMBCD could be used as the first part of calculating an average.



5

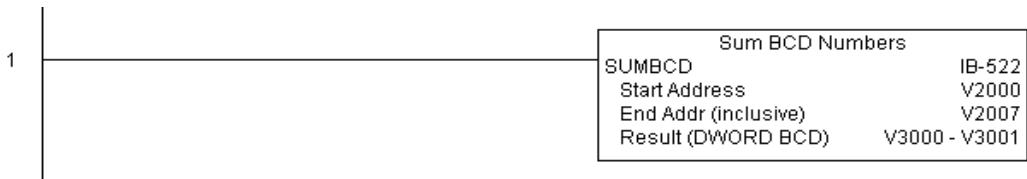
#### SUMBCD Parameters

- Start Address: specifies the starting address of a block of V-memory location values to be added together (BCD)
- End Addr (inclusive): specifies the ending address of a block of V-memory location values to be added together (BCD)
- Result (DWORD BCD): specifies the location where the sum of the block of V-memory BCD values will be placed

Parameter	DL05 Range
Start Address ..... V	See DL05 V-memory map - Data Words
End Address (inclusive) ..... V	See DL05 V-memory map - Data Words
Result (DWORD BCD) ..... V	See DL05 V-memory map - Data Words

#### SUMBCD Example

In the following example, the SUMBCD instruction is used to total the sum of all BCD values in words V2000 thru V2007 and store the resulting 8-digit double word BCD value in V3000 and V3001.



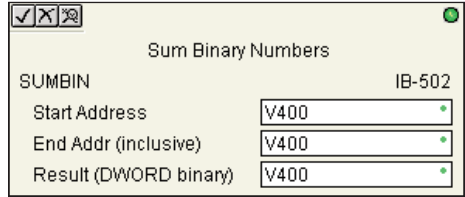
### Sum Binary Numbers (SUMBIN) (IB-502)

DS5	Used
HPP	N/A

Sum Binary Numbers sums up a list of consecutive 16-bit WORD Binary numbers into a 32-bit DWORD binary result.

You specify the group's starting and ending V-memory addresses (inclusive). When enabled, this instruction will add up all the numbers in the group (so you may want to place a differential contact driving the enable).

SUMBIN could be used as the first part of calculating an average.



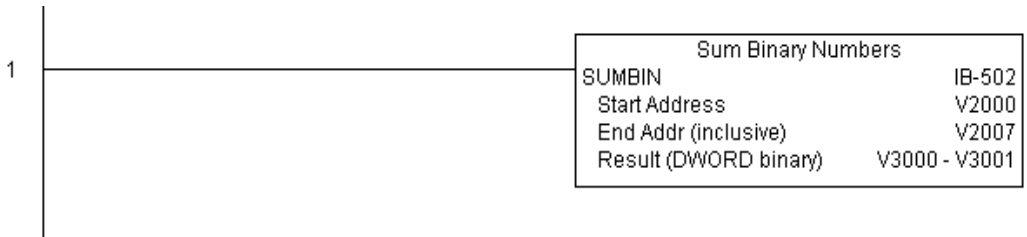
#### SUMBIN Parameters

- Start Address: specifies the starting address of a block of V-memory location values to be added together (Binary)
- End Addr (inclusive): specifies the ending address of a block of V-memory location values to be added together (Binary)
- Result (DWORD Binary): specifies the location where the sum of the block of V-memory binary values will be placed

Parameter	DL05 Range
Start Address . . . . . V	See DL05 V-memory map - Data Words
End Address (inclusive) . . . . . V	See DL05 V-memory map - Data Words
Result (DWORD Binary) . . . . . V	See DL05 V-memory map - Data Words

#### SUMBIN Example

In the following example, the SUMBIN instruction is used to total the sum of all Binary values in words V2000 thru V2007 and store the resulting 8-digit double word Binary value in V3000 and V3001.



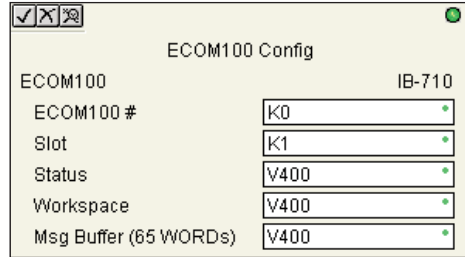
### ECOM100 Configuration (ECOM100) (IB-710)

DS5	Used
HPP	N/A

ECOM100 Configuration defines all the common information for one specific ECOM100 module which is used by the other ECOM100 IBoxes; for example, ECRX - ECOM100 Network Read , ECEMAIL - ECOM100 Send EMail, ECIPSUP - ECOM100 IP Setup, etc.

You MUST have the ECOM100 Configuration IBox at the top of your ladder/stage program with any other configuration IBoxes. The Message Buffer parameter specifies the starting address of a 65 WORD buffer. This is 101 Octal addresses (e.g. V1400 thru V1500).

If you have more than one ECOM100 in your PLC, you must have a different ECOM100 Configuration IBox for EACH ECOM100 module in your system that utilizes any ECOM IBox instructions.



The Workspace and Status parameters and the entire Message Buffer are internal, private registers used by the ECOM100 Configuration IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

In order for MOST ECOM100 IBoxes to function, you must turn ON dip switch 7 on the ECOM100 circuit board. You can keep dip switch 7 off if you are ONLY using ECOM100 Network Read and Write IBoxes (ECRX, ECWX).

#### ECOM100 Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Slot: specifies the option slot the module occupies
- Status: specifies a V-memory location that will be used by the instruction
- Workspace: specifies a V-memory location that will be used by the instruction
- Msg Buffer: specifies the starting address of a 65 word buffer that will be used by the module for configuration

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Slot ..... K	K1-4
Status ..... V	See DL05 V-memory map - Data Words
Workspace ..... V	See DL05 V-memory map - Data Words
Msg Buffer (65 words used) ..... V	See DL05 V-memory map - Data Words



### ECOM100 Example

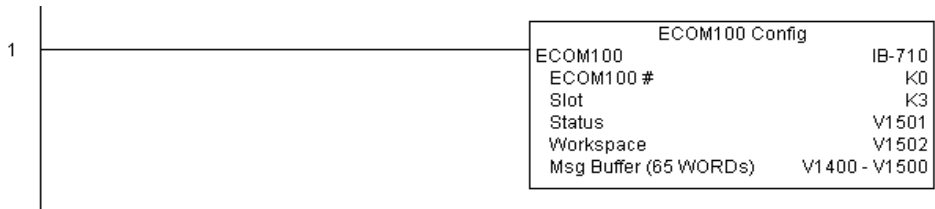
The ECOM100 Config IBox coordinates all of the interaction with other ECOM100 based IBoxes (ECxxxx). You must have an ECOM100 Config IBox for each ECOM100 module in your system. Configuration IBoxes must be at the top of your program and must execute every scan.

This IBox defines ECOM100# K0 to be in slot 3. Any ECOM100 IBoxes that need to reference this specific module (such as ECEMAIL, ECRX, ...) would enter K0 for their ECOM100# parameter.

The Status register is for reporting any completion or error information to other ECOM100 IBoxes. This V-memory register must not be used anywhere else in the entire program.

The Workspace register is used to maintain state information about the ECOM100, along with proper sharing and interlocking with the other ECOM100 IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.

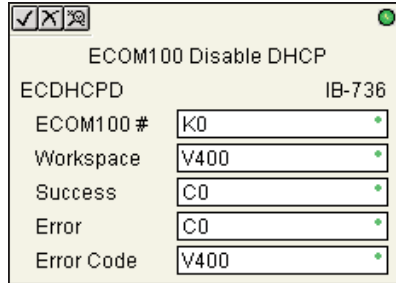
The Message Buffer of 65 words (130 bytes) is a common pool of memory that is used by other ECOM100 IBoxes (such as ECEMAIL). This way, you can have a bunch of ECEMAIL IBoxes, but only need 1 common buffer for generating and sending each EMail. These V-memory registers must not be used anywhere else in your entire program.



### ECOM100 Disable DHCP (ECDHCPD) (IB-736)

DS5	Used
HPP	N/A

ECOM100 Disable DHCP will setup the ECOM100 to use its internal TCP/IP settings on a leading edge transition to the IBox. To configure the ECOM100's TCP/IP settings manually, use the NetEdit3 utility, or you can do it programmatically from your PLC program using the ECOM100 IP Setup (ECIPSUP), or the individual ECOM100 IBoxes: ECOM Write IP Address (ECWRIP), ECOM Write Gateway Address (ECWRGWA), and ECOM100 Write Subnet Mask (ECWRSNM).



The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The "Disable DHCP" setting is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE**, on second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED** SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

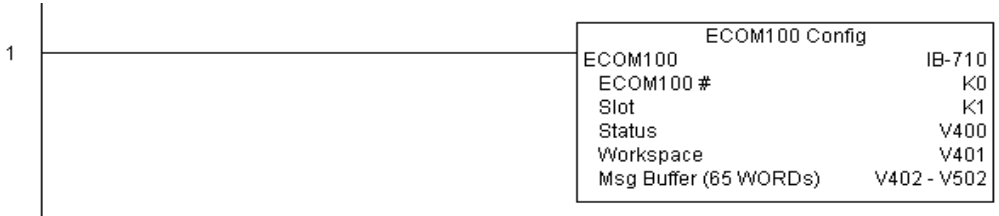
#### ECDHCPD Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error Code ..... V	See DL05 V-memory map - Data Words

### ECDHCPD Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



5

Rung 2: On the 2nd scan, disable DHCP in the ECOM100. DHCP is the same protocol used by PCs for using a DHCP Server to automatically assign the ECOM100's IP Address, Gateway Address, and Subnet Mask. Typically disabling DHCP is done by assigning a hard-coded IP Address either in NetEdit or using one of the ECOM100 IP Setup IBoxes, but this IBox allows you to disable DHCP in the ECOM100 using your ladder program. The ECDHCPD is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to disable DHCP will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON. If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



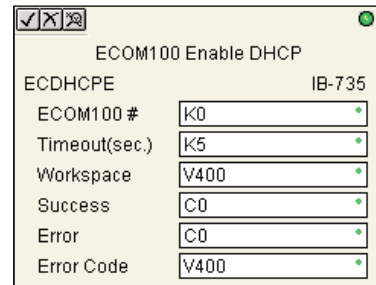
### ECOM100 Enable DHCP (ECDHCPE) (IB-735)

ECOM100 Enable DHCP will tell the ECOM100 to obtain its TCP/IP setup from a DHCP Server on a leading edge transition to the IBox.

DS5	Used
HPP	N/A

The IBox will be successful once the ECOM100 has received its TCP/IP settings from the DHCP server. Since it is possible for the DHCP server to be unavailable, a Timeout parameter is provided so the IBox can complete, but with an Error (Error Code = 1004 decimal).

See also the ECOM100 IP Setup (ECIPSUP) IBox 717 to directly setup ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.



The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The "Enable DHCP" setting is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is HIGHLY RECOMMENDED that you only execute this IBox ONCE, on second scan. Since it requires a LEADING edge to execute, use a NORMALLY CLOSED SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

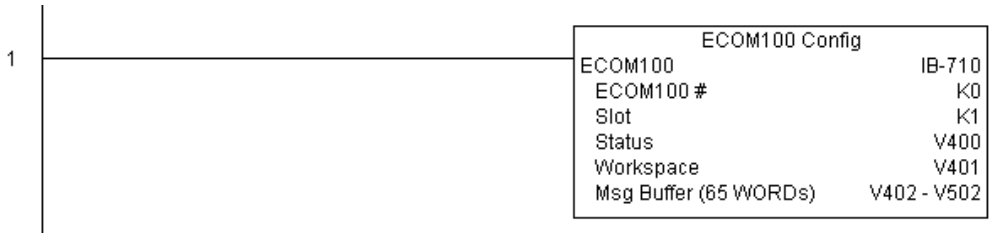
#### ECDHCPE Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Timeout(sec): specifies a timeout period so that the instruction may have time to complete
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written

Parameter	DL05 Range
ECOM100# . . . . . K	K0-255
Timeout (sec) . . . . . K	K5-127
Workspace . . . . . V	See DL05 V-memory map - Data Words
Success . . . . . X,Y,C,GX,GY,B	See DL05 V-memory map
Error . . . . . X,Y,C,GX,GY,B	See DL05 V-memory map
Error Code . . . . . V	See DL05 V-memory map - Data Words

### ECDHCPE Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



Rung 2: On the 2nd scan, enable DHCP in the ECOM100. DHCP is the same protocol used by PCs for using a DHCP Server to automatically assign the ECOM100's IP Address, Gateway Address, and Subnet Mask. Typically this is done using NetEdit, but this IBox allows you to enable DHCP in the ECOM100 using your ladder program. The ECDHCPE is leading edge triggered, not power-flow driven (similar to a counter input leg). The commands to enable DHCP will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON. The ECDHCPE does more than just set the bit to enable DHCP in the ECOM100, but it then polls the ECOM100 once every second to see if the ECOM100 has found a DHCP server and has a valid IP Address. Therefore, a timeout parameter is needed in case the ECOM100 cannot find a DHCP server. If a timeout does occur, the Error bit will turn on and the error code will be 1005 decimal. The Success bit will turn on only if the ECOM100 finds a DHCP Server and is assigned a valid IP Address. If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



### ECOM100 Query DHCP Setting (ECDHCPQ) (IB-734)

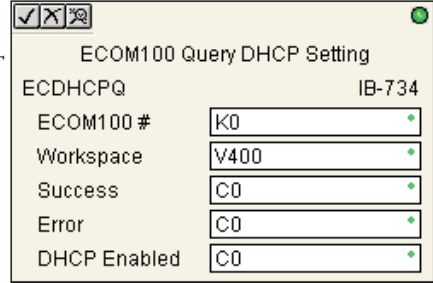
DS5	Used
HPP	N/A

ECOM100 Query DHCP Setting will determine if DHCP is enabled in the ECOM100 on a leading edge transition to the IBox. The DHCP Enabled bit parameter will be ON if DHCP is enabled, OFF if disabled.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.



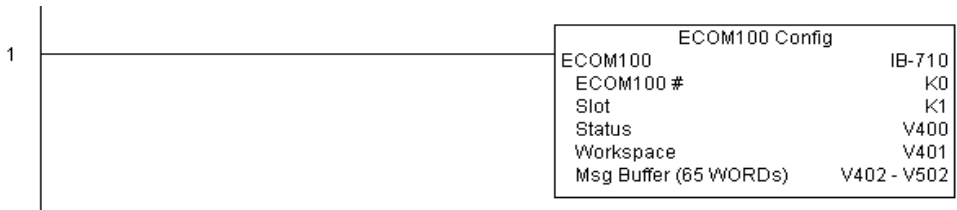
#### ECDHCPQ Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- DHCP Enabled: specifies a bit that will turn on if the ECOM100's DHCP is enabled or remain off if disabled - after instruction query, be sure to check the state of the Success/Error bit state along with DHCP Enabled bit state to confirm a successful module query

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map
DHCP Enabled ..... X,Y,C,GX,GY,B	See DL05 V-memory map

### ECDHCPQ Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



5

Rung 2: On the 2nd scan, read whether DHCP is enabled or disabled in the ECOM100 and store it in C5. DHCP is the same protocol used by PCs for using a DHCP Server to automatically assign the ECOM100's IP Address, Gateway Address, and Subnet Mask. The ECDHCPQ is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read (Query) whether DHCP is enabled or not will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON. If successful, turn on C100. If there is a failure, turn on C101.



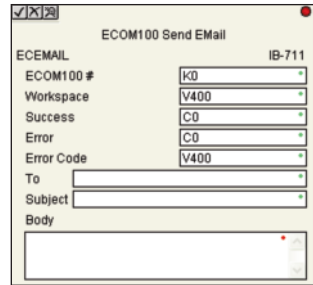
## ECOM100 Send E-mail (ECEMAIL) (IB-711)

DS5	Used
HPP	N/A

ECOM100 Send EMail, on a leading edge transition, will behave as an EMail client and send an SMTP request to your SMTP Server to send the EMail message to the EMail addresses in the To: field and also to those listed in the Cc: list hard coded in the ECOM100. It will send the SMTP request based on the specified ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) at the top of your program.

The Body: field supports what the PRINT and VPRINT instructions support for text and embedded variables, allowing you to embed real-time data in your EMail (e.g. "V2000 = " V2000:B).

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.



Either the Success or Error bit parameter will turn on once the request is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), an SMTP protocol error (between 100 and 999), or a PLC logic error (greater than 1000).

Since the ECOM100 is only an EMail Client and requires access to an SMTP Server, you **MUST** have the SMTP parameters configured properly in the ECOM100 via the ECOM100's Home Page and/or the EMail Setup instruction (ECEMSUP). To get to the ECOM100's Home Page, use your favorite Internet browser and browse to the ECOM100's IP Address, e.g. <http://192.168.12.86>

You are limited to approximately 100 characters of message data for the entire instruction, including the To: Subject: and Body: fields. To save space, the ECOM100 supports a hard coded list of EMail addresses for the Carbon Copy field (cc:) so that you can configure those IN the ECOM100, and keep the To: field small (or even empty), to leave more room for the Subject: and Body: fields.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

### ECEMAIL Parameters

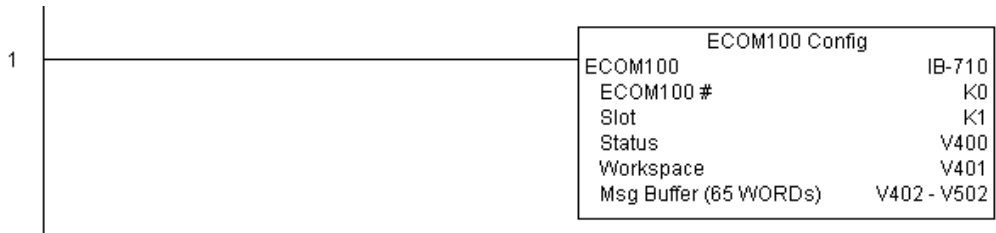
- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written
- To: specifies an E-mail address that the message will be sent to
- Subject: subject of the e-mail message
- Body: supports what the PRINT and VPRINT instructions support for text and embedded variables, allowing you to embed real-time data in the EMail message



Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error Code ..... V	See DL05 V-memory map
To:.....	Text
Subject:.....	Text
Body:.....	See PRINT and VPRINT instructions

**ECEMAIL Example**

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



(example continued on next page)

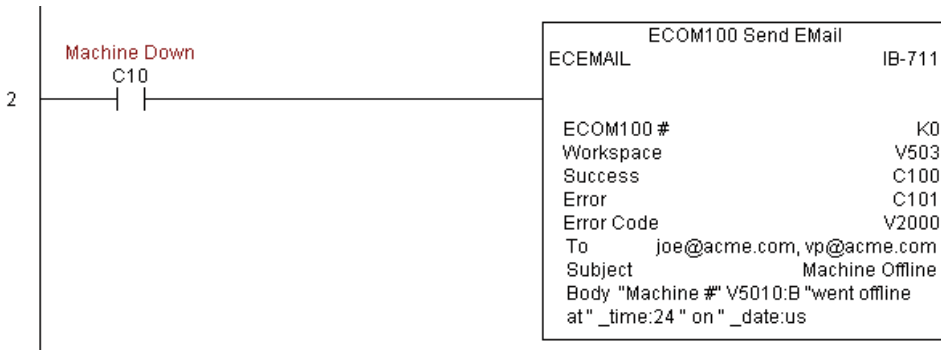
### ECEMAIL Example (cont'd)

Rung 2: When a machine goes down, send an email to Joe in maintenance and to the VP over production showing what machine is down along with the date/time stamp of when it went down.

The ECEMAIL is leading edge triggered, not power-flow driven (similar to a counter input leg). An email will be sent whenever the power flow into the IBox goes from OFF to ON. This helps prevent self inflicted spamming.

If the EMail is sent, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the SMTP error code or other possible error codes.

5



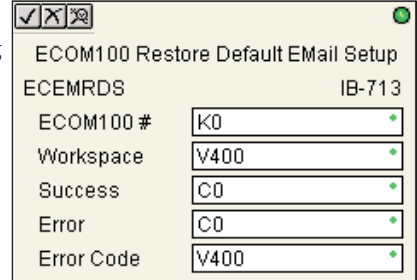
### ECOM100 Restore Default E-mail Setup (ECEMRDS) (IB-713)

DS5	Used
HPP	N/A

ECOM100 Restore Default EMail Setup, on a leading edge transition, will restore the original EMail Setup data stored in the ECOM100 back to the working copy based on the specified ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) at the top of your program.

When the ECOM100 is first powered up, it copies the EMail setup data stored in ROM to the working copy in RAM. You can then modify this working copy from your program using the ECOM100 EMail Setup (ECEMSUP) IBox. After modifying the working copy, you can later restore the original setup data via your program by using this IBox.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.



Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

#### ECEMRDS Parameters

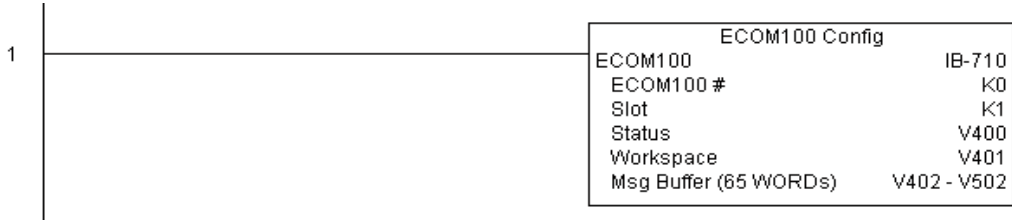
- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error Code ..... V	See DL05 V-memory map - Data Words

### ECEMRDS Example

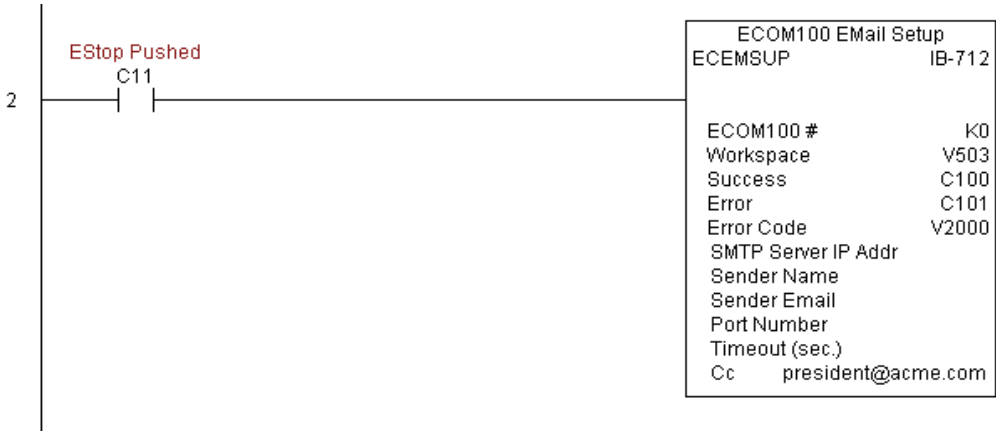
Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5



Rung 2: Whenever an EStop is pushed, ensure that president of the company gets copies of all EMail being sent.

The ECOM100 EMail Setup IBox allows you to set/change the SMTP EMail settings stored in the ECOM100.

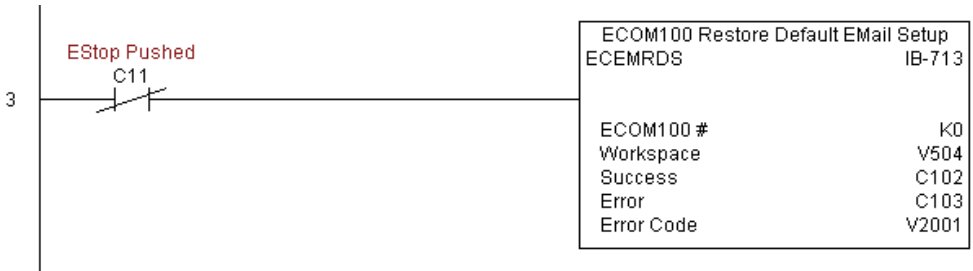


### ECEMRDS Example

Rung 3: Once the EStop is pulled out, take the president off the cc: list by restoring the default EMail setup in the ECOM100.

The ECEMRDS is leading edge triggered, not power-flow driven (similar to a counter input leg). The ROM based EMail configuration stored in the ECOM100 will be copied over the "working copy" whenever the power flow into the IBox goes from OFF to ON (the working copy can be changed by using the ECEMSUP IBox).

If successful, turn on C102. If there is a failure, turn on C103. If it fails, you can look at V2001 for the specific error code.

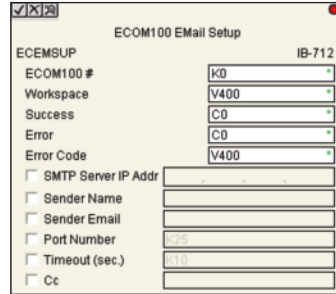


## ECOM100 E-mail Setup (ECEMSUP) (IB-712)

DS5	Used
HPP	N/A

ECOM100 EMail Setup, on a leading edge transition, will modify the working copy of the EMail setup currently in the ECOM100 based on the specified ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) at the top of your program.

You may pick and choose any or all fields to be modified using this instruction. Note that these changes are cumulative: if you execute multiple ECOM100 EMail Setup IBoxes, then all of the changes are made in the order they are executed. Also note that you can restore the original ECOM100 EMail Setup that is stored in the ECOM100 to the working copy by using the ECOM100 Restore Default EMail Setup (ECEMRDS) IBox.



The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

You are limited to approximately 100 characters/bytes of setup data for the entire instruction. So if needed, you could divide the entire setup across multiple ECEMSUP IBoxes on a field-by-field basis, for example do the Carbon Copy (cc:) field in one ECEMSUP IBox and the remaining setup parameters in another.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

### ECEMSUP Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written
- SMTP Server IP Addr: optional parameter that specifies the IP Address of the SMTP Server on the ECOM100's network
- Sender Name: optional parameter that specifies the sender name that will appear in the "From:" field to those who receive the e-mail
- Sender EMail: optional parameter that specifies the sender EMail address that will appear in the "From:" field to those who receive the e-mail

**ECEMSUP Parameters**

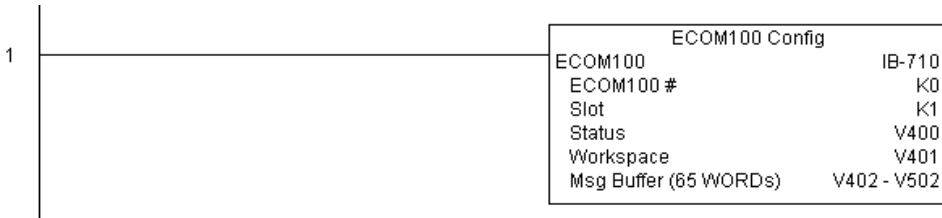
- Port Number: optional parameter that specifies the TCP/IP Port Number to send SMTP requests; usually this does not to be configured (see your network administrator for information on this setting)
- Timeout (sec): optional parameter that specifies the number of seconds to wait for the SMTP Server to send the EMail to all the recipients
- Cc: optional parameter that specifies a list of “carbon copy” Email addresses to send all EMails to

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error Code ..... V	See DL05 V-memory map - Data Words

### ECEMSUP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5

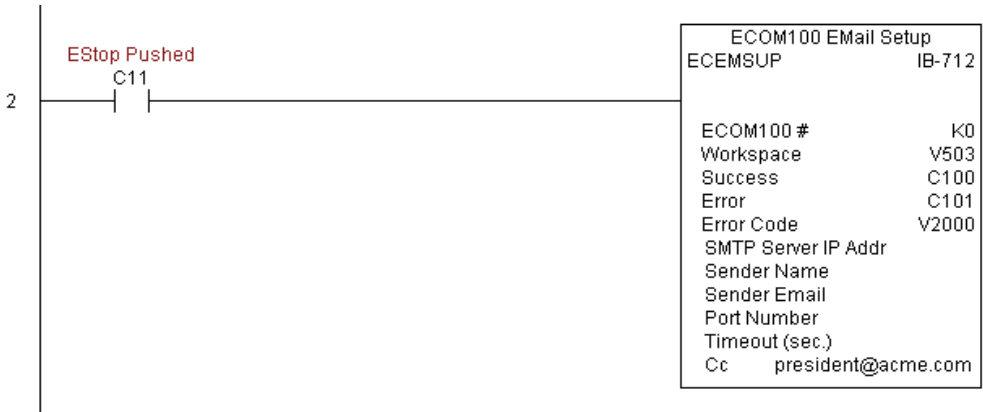




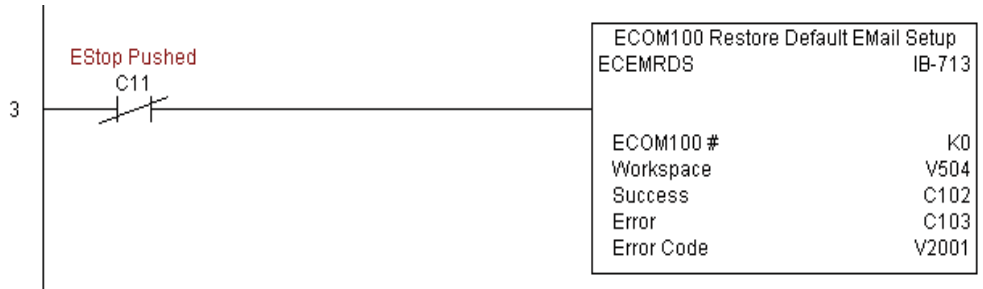
### ECEMSUP Example

Rung 2: Whenever an EStop is pushed, ensure that president of the company gets copies of all EMail being sent. The ECOM100 EMail Setup IBox allows you to set/change the SMTP EMail settings stored in the ECOM100. The ECEMSUP is leading edge triggered, not power-flow driven (similar to a counter input leg). At power-up, the ROM based EMail configuration stored in the ECOM100 is copied to a RAM based "working copy". You can change this working copy by using the ECEMSUP IBox. To restore the original ROM based configuration, use the Restore Default EMail Setup ECEMRDS IBox.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



Rung 3: Once the EStop is pulled out, take the president off the cc: list by restoring the default EMail setup in the ECOM100.

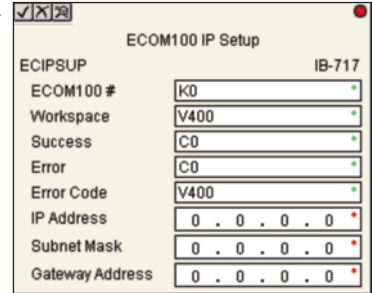


### ECOM100 IP Setup (ECIPSUP) (IB-717)

DS5	Used
HPP	N/A

ECOM100 IP Setup will configure the three TCP/IP parameters in the ECOM100: IP Address, Subnet Mask, and Gateway Address, on a leading edge transition to the IBox. The ECOM100 is specified by the ECOM100#, which corresponds to a specific unique ECOM100 Configuration (ECOM100) IBox at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.



Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

This setup data is stored in Flash-ROM in the ECOM100 and will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED SP0 (NOT First Scan)** to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

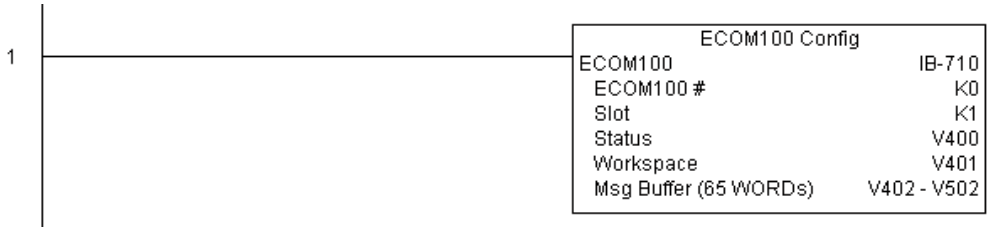
#### ECIPSUP Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written
- IP Address: specifies the module's IP Address
- Subnet Mask: specifies the Subnet Mask for the module to use
- Gateway Address: specifies the Gateway Address for the module to use

Parameter	DL05 Range
ECOM100# . . . . . K	K0-255
Workspace . . . . . V	See DL05 V-memory map - Data Words
Success . . . . . X,Y,C,GX,GY,B	See DL05 V-memory map
Error . . . . . X,Y,C,GX,GY,B	See DL05 V-memory map
Error Code . . . . . V	See DL05 V-memory map - Data Words
IP Address . . . . . IP Address	0.0.0.1. to 255.255.255.254
Subnet Mask Address . . . . . IP Address Mask	0.0.0.1. to 255.255.255.254
Gateway Address . . . . . IP Address	0.0.0.1. to 255.255.255.254

### ECIPSUP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

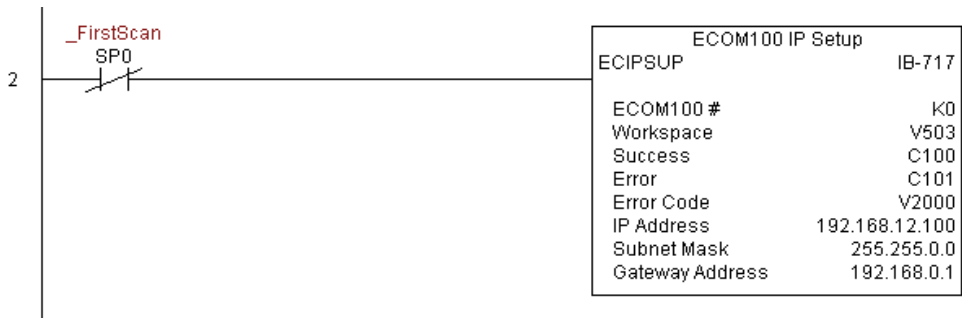


Rung 2: On the 2nd scan, configure all of the TCP/IP parameters in the ECOM100:

IP Address: 192.168. 12.100  
 Subnet Mask: 255.255. 0. 0  
 Gateway Address: 192.168. 0. 1

The ECIPSUP is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the TCP/IP configuration parameters will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



### ECOM100 Read Description (ECRDDDES) (IB-726)

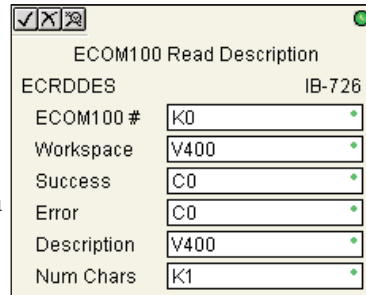
DS5	Used
HPP	N/A

ECOM100 Read Description will read the ECOM100's Description field up to the number of specified characters on a leading edge transition to the IBox.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.



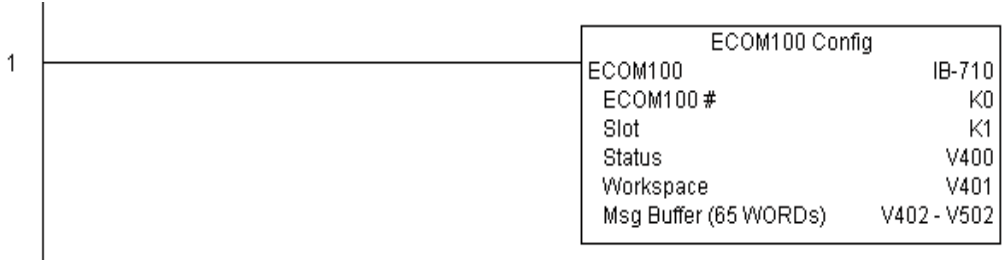
#### ECRDDDES Parameters

- **ECOM100#:** this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- **Workspace:** specifies a V-memory location that will be used by the instruction
- **Success:** specifies a bit that will turn on once the request is completed successfully
- **Error:** specifies a bit that will turn on if the instruction is not successfully completed
- **Description:** specifies the starting buffer location where the ECOM100's Module Name will be placed
- **Num Char:** specifies the number of characters (bytes) to read from the ECOM100's Description field

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Description ..... V	See DL05 V-memory map - Data Words
Num Chars ..... K	K1-128

### ECRDDES Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



5

Rung 2: On the 2nd scan, read the Module Description of the ECOM100 and store it in V3000 thru V3007 (16 characters). This text can be displayed by an HMI.

The ECRDDES is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the module description will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



### ECOM100 Read Gateway Address (ECRDGWA) (IB-730)

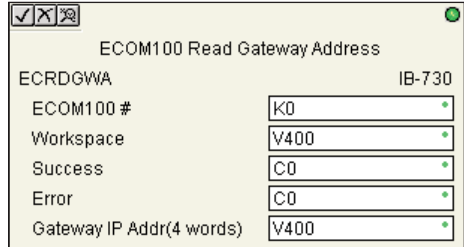
DS5	Used
HPP	N/A

ECOM100 Read Gateway Address will read the 4 parts of the Gateway IP address and store them in 4 consecutive V-memory locations in decimal format, on a leading edge transition to the IBox.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.



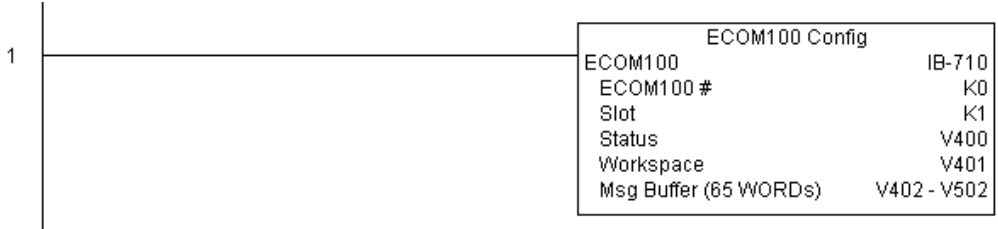
#### ECRDGWA Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Gateway IP Addr: specifies the starting address where the ECOM100's Gateway Address will be placed in 4 consecutive V-memory locations

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Gateway IP Address (4 Words) ..... V	See DL05 V-memory map - Data Words

### ECRDGWA Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

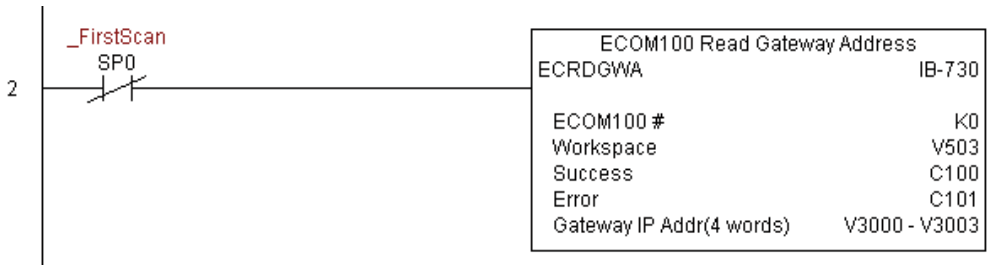


5

Rung 2: On the 2nd scan, read the Gateway Address of the ECOM100 and store it in V3000 thru V3003 (4 decimal numbers). The ECOM100's Gateway Address could be displayed by an HMI.

The ECRDGWA is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the Gateway Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



### ECOM100 Read IP Address (ECRDIP) (IB-722)

DS5	Used
HPP	N/A

ECOM100 Read IP Address will read the 4 parts of the IP address and store them in 4 consecutive V-memory locations in decimal format, on a leading edge transition to the IBox.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

#### ECRDIP Parameters

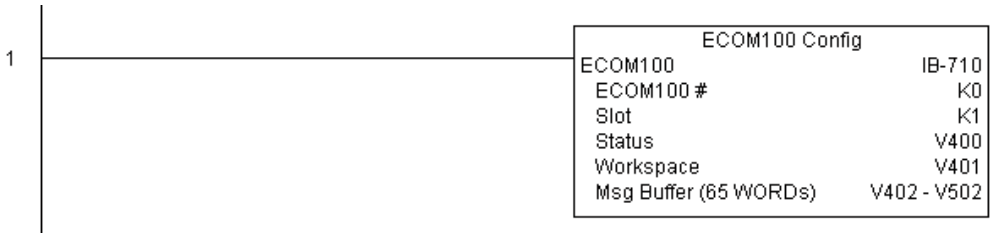
- **ECOM100#:** this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- **Workspace:** specifies a V-memory location that will be used by the instruction
- **Success:** specifies a bit that will turn on once the request is completed successfully
- **Error:** specifies a bit that will turn on if the instruction is not successfully completed
- **IP Address:** specifies the starting address where the ECOM100's IP Address will be placed in 4 consecutive V-memory locations

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map
IP Address (4 Words) ..... V	See DL05 V-memory map - Data Words



### ECRDIP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

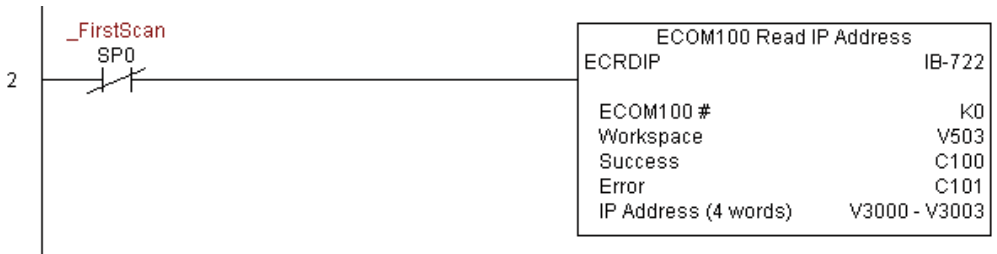


5

Rung 2: On the 2nd scan, read the IP Address of the ECOM100 and store it in V3000 thru V3003 (4 decimal numbers). The ECOM100's IP Address could be displayed by an HMI.

The ECRDIP is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the IP Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



**ECOM100 Read Module ID (ECRDMID) (IB-720)**

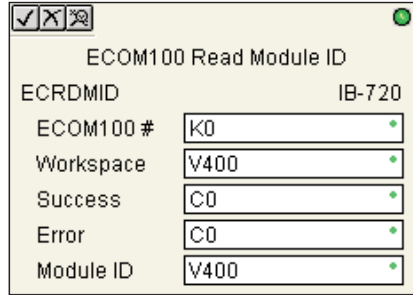
DS5	Used
HPP	N/A

ECOM100 Read Module ID will read the binary (decimal) WORD sized Module ID on a leading edge transition to the IBox.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.



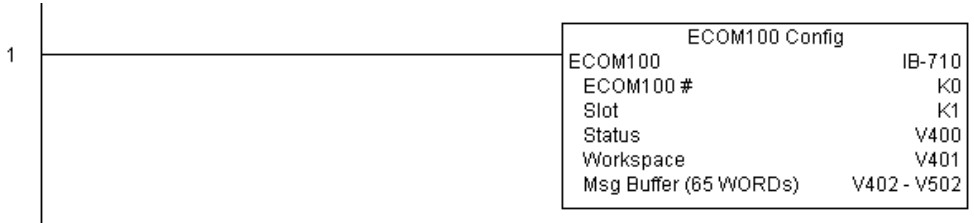
**ECRDMID Parameters**

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Module ID: specifies the location where the ECOM100's Module ID (decimal) will be placed

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Module ID..... V	See DL05 V-memory map - Data Words

### ECRDMID Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



5

Rung 2: On the 2nd scan, read the Module ID of the ECOM100 and store it in V2000. The ECRDMID is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the module ID will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON. If successful, turn on C100. If there is a failure, turn on C101.



**ECOM100 Read Module Name (ECRDNAM) (IB-724)**

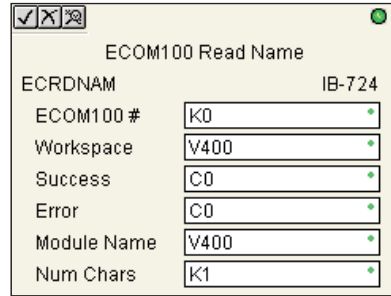
DS5	Used
HPP	N/A

ECOM100 Read Name will read the Module Name up to the number of specified characters on a leading edge transition to the IBox.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.



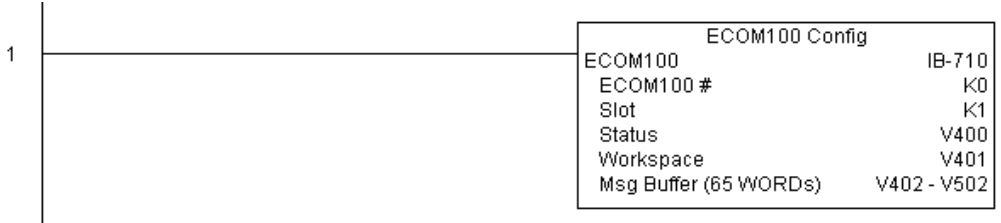
**ECRDNAM Parameters**

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Module Name: specifies the starting buffer location where the ECOM100's Module Name will be placed
- Num Chars: specifies the number of characters (bytes) to read from the ECOM100's Name field

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Module Name ..... V	See DL05 V-memory map - Data Words
Num Chars ..... K	K1-128

### ECRDNAM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

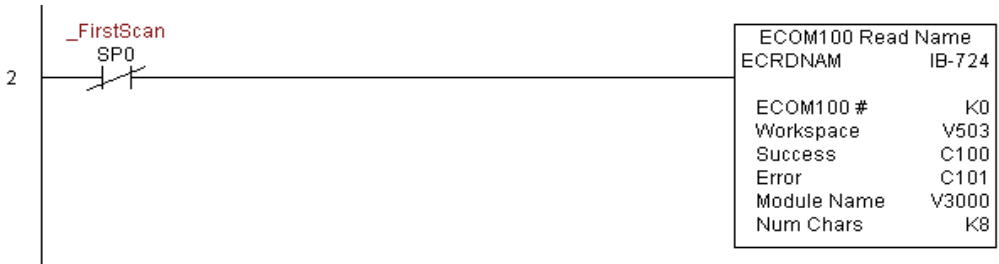


5

Rung 2: On the 2nd scan, read the Module Name of the ECOM100 and store it in V3000 thru V3003 (8 characters). This text can be displayed by an HMI.

The ECRDNAM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the module name will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



**ECOM100 Read Subnet Mask (ECRDSNM) (IB-732)**

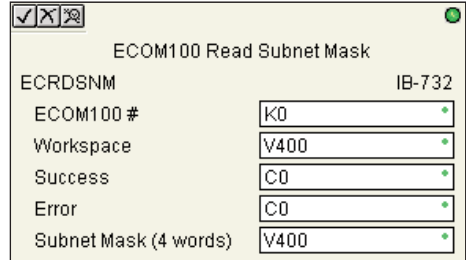
DS5	Used
HPP	N/A

ECOM100 Read Subnet Mask will read the 4 parts of the Subnet Mask and store them in 4 consecutive V-memory locations in decimal format, on a leading edge transition to the IBox.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.



5

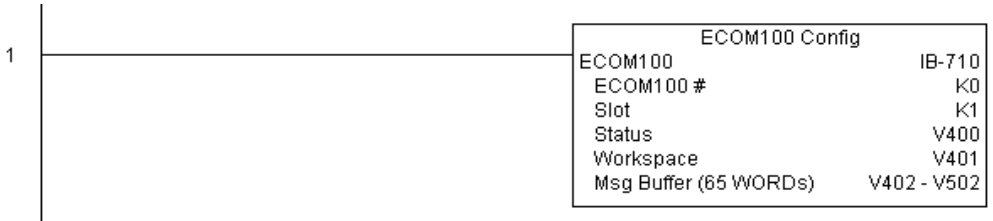
**ECRDSNM Parameters**

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Subnet Mask: specifies the starting address where the ECOM100's Subnet Mask will be placed in 4 consecutive V-memory locations

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Subnet Mask (4 Words)..... V	See DL05 V-memory map - Data Words

### ECRDSNM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

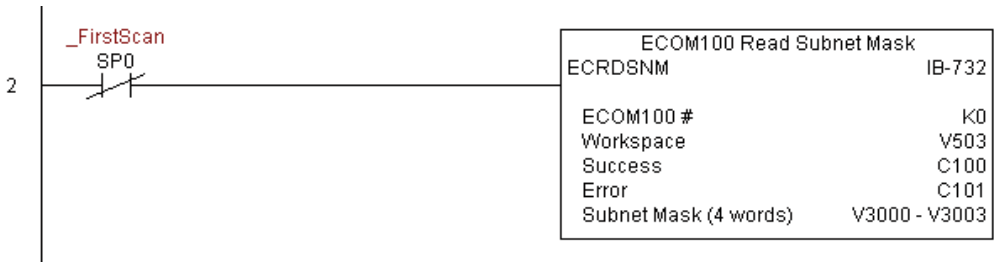


5

Rung 2: On the 2nd scan, read the Subnet Mask of the ECOM100 and store it in V3000 thru V3003 (4 decimal numbers). The ECOM100's Subnet Mask could be displayed by an HMI.

The ECRDSNM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to read the Subnet Mask will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101.



**ECOM100 Write Description (ECWRDES) (IB-727)**

DS5	Used
HPP	N/A

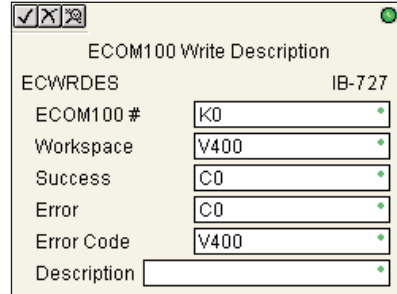
ECOM100 Write Description will write the given Description to the ECOM100 module on a leading edge transition to the IBox. If you use a dollar sign (\$) or double quote ("), use the PRINT/VPRINT escape sequence of TWO dollar signs (\$\$) for a single dollar sign or dollar sign-double quote (\$") for a double quote character.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Description is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED** SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.



**ECWRDES Parameters**

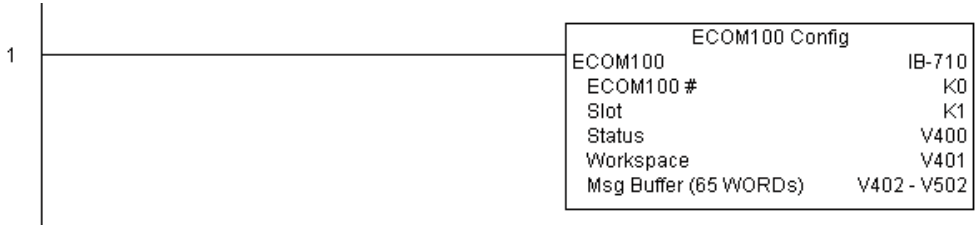
- **ECOM100#:** this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- **Workspace:** specifies a V-memory location that will be used by the instruction
- **Success:** specifies a bit that will turn on once the request is completed successfully
- **Error:** specifies a bit that will turn on if the instruction is not successfully completed
- **Error Code:** specifies the location where the Error Code will be written
- **Description:** specifies the Description that will be written to the module

Parameter	DL05 Range
ECOM100# .....	K
Workspace .....	V
Success .....	X,Y,C,GX,GY,B
Error .....	X,Y,C,GX,GY,B
Error Code .....	V
Description .....	Text



### ECWRDES Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

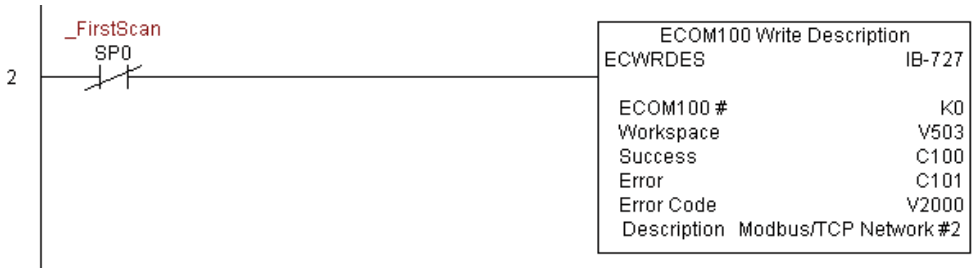


5

Rung 2: On the 2nd scan, set the Module Description of the ECOM100. Typically this is done using NetEdit, but this IBox allows you to configure the module description in the ECOM100 using your ladder program.

The EWRDES is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the module description will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



**ECOM100 Write Gateway Address (ECWRGWA) (IB-731)**

DS5	Used
HPP	N/A

ECOM100 Write Gateway Address will write the given Gateway IP Address to the ECOM100 module on a leading edge transition to the IBox. See also ECOM100 IP Setup (ECIPSUP) IBox 717 to setup ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Gateway Address is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE**, on second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED SP0 (STR NOT First Scan)** to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

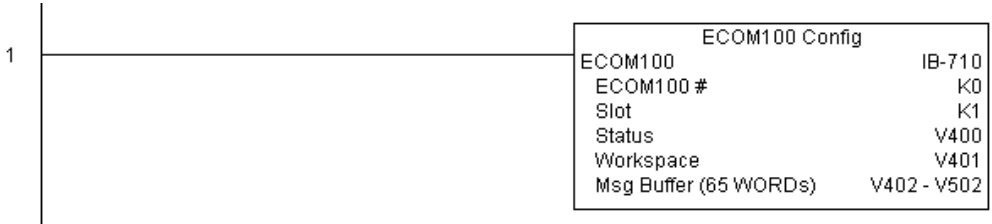
**ECWRGWA Parameters**

- **ECOM100#:** this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- **Workspace:** specifies a V-memory location that will be used by the instruction
- **Success:** specifies a bit that will turn on once the request is completed successfully
- **Error:** specifies a bit that will turn on if the instruction is not successfully completed
- **Error Code:** specifies the location where the Error Code will be written
- **Gateway Address:** specifies the Gateway IP Address that will be written to the module

Parameter	DL05 Range
ECOM100# .....	K0-255
Workspace .....	See DL05 V-memory map - Data Words
Success .....	See DL05 V-memory map
Error .....	See DL05 V-memory map
Error Code .....	See DL05 V-memory map - Data Words
Gateway Address .....	0.0.0.1. to 255.255.255.254

### ECWRGWA Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

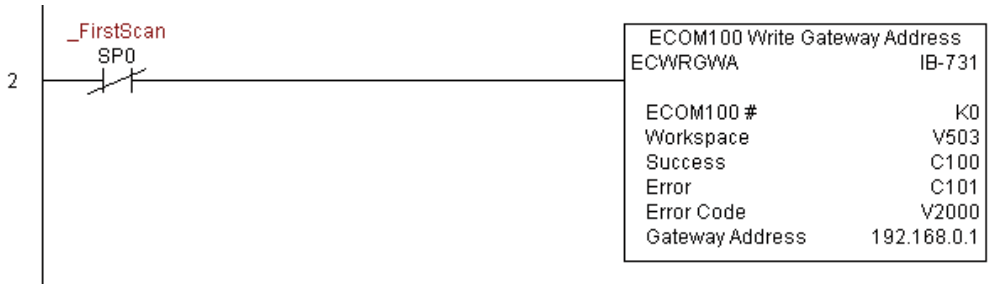


5

Rung 2: On the 2nd scan, assign the Gateway Address of the ECOM100 to 192.168.0.1  
 The ECWRGWA is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the Gateway Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

To configure all of the ECOM100 TCP/IP parameters in one IBox, see the ECOM100 IP Setup (ECIPSUP) IBox.



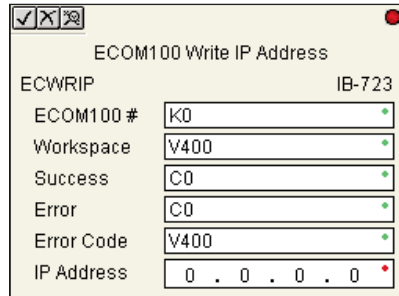
### ECOM100 Write IP Address (ECWRIP) (IB-723)

DS5	Used
HPP	N/A

ECOM100 Write IP Address will write the given IP Address to the ECOM100 module on a leading edge transition to the IBox. See also ECOM100 IP Setup (ECIPSUP) IBox 717 to setup ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).



The IP Address is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED** SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.

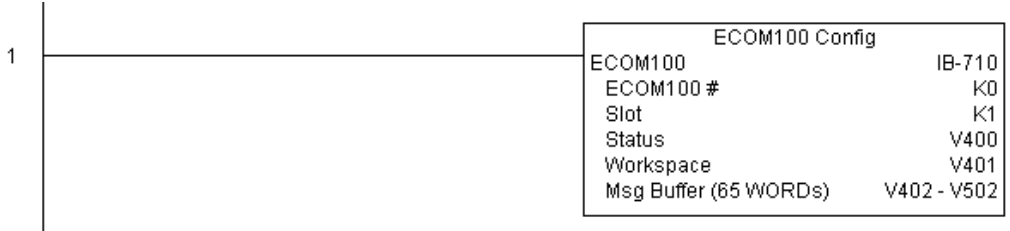
#### ECWRIP Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written
- IP Address: specifies the IP Address that will be written to the module

Parameter	DL05 Range
ECOM100#	K0-255
Workspace	See DL05 V-memory map - Data Words
Success	See DL05 V-memory map
Error	See DL05 V-memory map
Error Code	See DL05 V-memory map - Data Words
IP Address	0.0.0.1. to 255.255.255.254

### ECWRIP Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



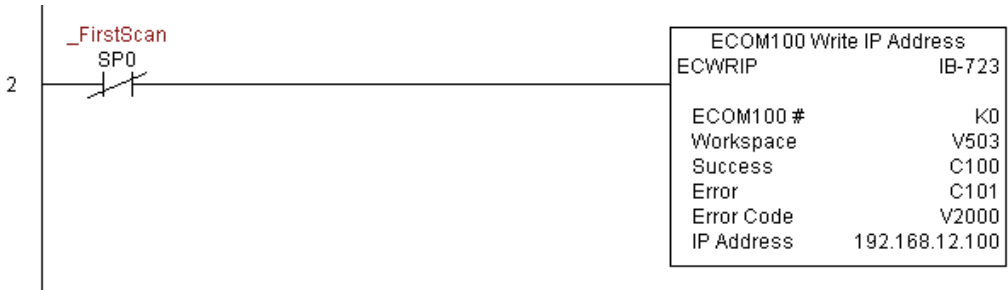
5

Rung 2: On the 2nd scan, assign the IP Address of the ECOM100 to 192.168.12.100

The ECWRIP is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the IP Address will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

To configure all of the ECOM100 TCP/IP parameters in one IBox, see the ECOM100 IP Setup (ECIPSUP) IBox.



**ECOM100 Write Module ID (ECWRMID) (IB-721)**

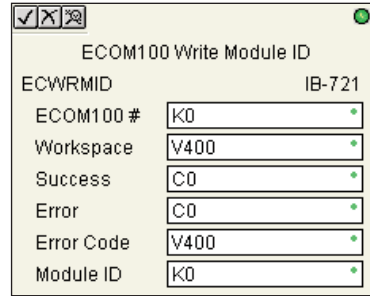
DS5	Used
HPP	N/A

ECOM100 Write Module ID will write the given Module ID on a leading edge transition to the IBox

If the Module ID is set in the hardware using the dipswitches, this IBox will fail and return error code 1005 (decimal).

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).



The Module ID is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED** SPO (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

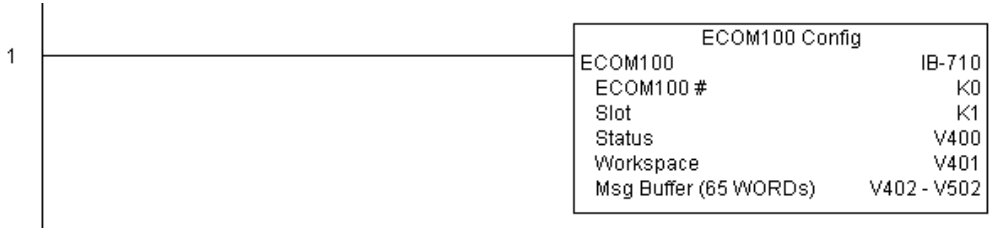
**ECWRMID Parameters**

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written
- Module ID: specifies the Module ID that will be written to the module

Parameter	DL05 Range
ECOM100# .....	K0-255
Workspace .....	See DL05 V-memory map - Data Words
Success .....	See DL05 V-memory map
Error .....	See DL05 V-memory map
Error Code .....	See DL05 V-memory map - Data Words
Module ID .....	K0-65535

### ECWRMID Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



5

Rung 2: On the 2nd scan, set the Module ID of the ECOM100. Typically this is done using NetEdit, but this IBox allows you to configure the module ID of the ECOM100 using your ladder program.

The EWRMID is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the module ID will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



**ECOM100 Write Name (ECWRNAM) (IB-725)**

DS5	Used
HPP	N/A

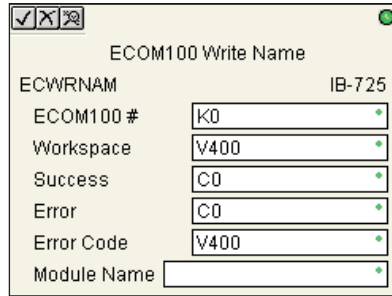
ECOM100 Write Name will write the given Name to the ECOM100 module on a leading edge transition to the IBox. If you use a dollar sign (\$) or double quote ("), use the PRINT/VPRINT escape sequence of TWO dollar signs (\$\$) for a single dollar sign or dollar sign-double quote (\$") for a double quote character.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).

The Name is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED** SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn ON dip switch 7 on the ECOM100 circuit board.



**ECWRNAM Parameters**

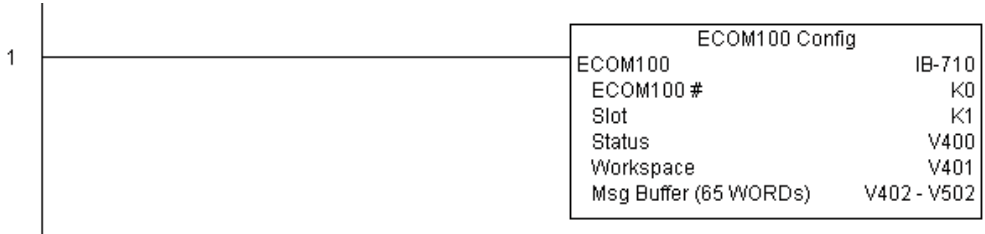
- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written
- Module Name: specifies the Name that will be written to the module

Parameter	DL05 Range
ECOM100# .....	K
Workspace .....	V
Success .....	X,Y,C,GX,GY,B
Error .....	X,Y,C,GX,GY,B
Error Code .....	V
Module Name .....	Text



### ECWRNAM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



5

Rung 2: On the 2nd scan, set the Module Name of the ECOM100. Typically this is done using NetEdit, but this IBox allows you to configure the module name of the ECOM100 using your ladder program.

The EWRNAM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the module name will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.



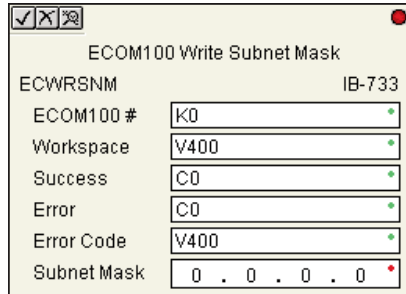
### ECOM100 Write Subnet Mask (ECWRSNM) (IB-733)

DS5	Used
HPP	N/A

ECOM100 Write Subnet Mask will write the given Subnet Mask to the ECOM100 module on a leading edge transition to the IBox. See also ECOM100 IP Setup (ECIPSUP) IBox 717 to setup ALL of the TCP/IP parameters in a single instruction - IP Address, Subnet Mask, and Gateway Address.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Either the Success or Error bit parameter will turn on once the command is complete. If there is an error, the Error Code parameter will report an ECOM100 error code (less than 100), or a PLC logic error (greater than 1000).



The Subnet Mask is stored in Flash-ROM in the ECOM100 and the execution of this IBox will disable the ECOM100 module for at least a half second until it writes the Flash-ROM. Therefore, it is **HIGHLY RECOMMENDED** that you only execute this IBox **ONCE** on second scan. Since it requires a **LEADING** edge to execute, use a **NORMALLY CLOSED** SP0 (STR NOT First Scan) to drive the power flow to the IBox.

In order for this ECOM100 IBox to function, you must turn **ON** dip switch 7 on the ECOM100 circuit board.

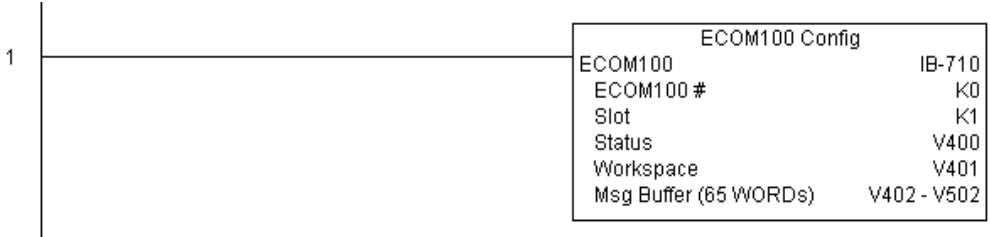
#### ECWRSNM Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed
- Error Code: specifies the location where the Error Code will be written
- Subnet Mask: specifies the Subnet Mask that will be written to the module

Parameter	DL05 Range
ECOM100# .....	K
ECOM100# .....	K0-255
Workspace .....	V
Workspace .....	See DL05 V-memory map - Data Words
Success .....	X,Y,C,GX,GY,B
Success .....	See DL05 V-memory map
Error .....	X,Y,C,GX,GY,B
Error .....	See DL05 V-memory map
Error Code .....	V
Error Code .....	See DL05 V-memory map - Data Words
Subnet Mask .....	
Subnet Mask .....	Masked IP Address

### ECWRSNM Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



Rung 2: On the 2nd scan, assign the Subnet Mask of the ECOM100 to 255.255.0.0

The ECWRSNM is leading edge triggered, not power-flow driven (similar to a counter input leg). The command to write the Subnet Mask will be sent to the ECOM100 whenever the power flow into the IBox goes from OFF to ON.

If successful, turn on C100. If there is a failure, turn on C101. If it fails, you can look at V2000 for the specific error code.

To configure all of the ECOM100 TCP/IP parameters in one IBox, see the ECOM100 IP Setup (ECIPSUP) IBox.



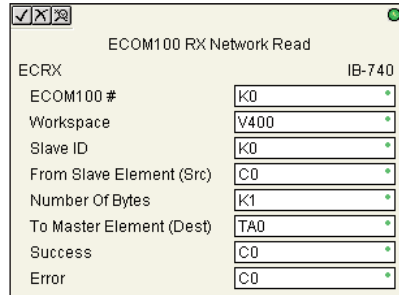
### ECOM100 RX Network Read (ECRX) (IB-740)

DS5	Used
HPP	N/A

ECOM100 RX Network Read performs the RX instruction with built-in interlocking with all other ECOM100 RX (ECRX) and ECOM100 WX (ECWX) IBoxes in your program to simplify communications networking. It will perform the RX on the specified ECOM100#’s network, which corresponds to a specific unique ECOM100 Configuration (ECOM100) IBox at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

Whenever this IBox has power, it will read element data from the specified slave into the given destination V-memory buffer, giving other ECOM100 RX and ECOM100 WX IBoxes on that ECOM100# network a chance to execute.



For example, if you wish to read and write data continuously from 5 different slaves, you can have all of these ECRX and ECWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically.

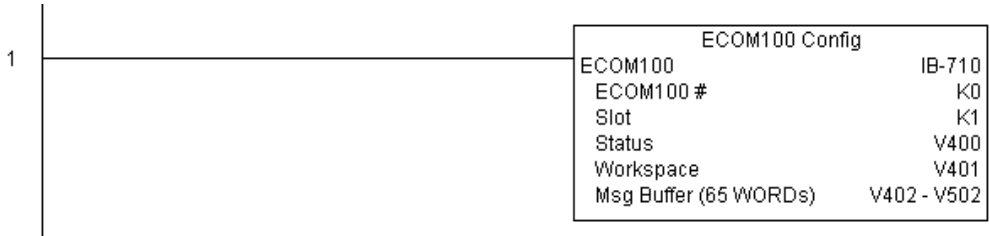
#### ECRX Parameters

- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Slave ID: specifies the slave ECOM(100) PLC that will be targeted by the ECRX instruction
- From Slave Element (Src): specifies the slave address of the data to be read
- Number of Bytes: specifies the number of bytes to read from the slave ECOM(100) PLC
- To Master Element (Dest): specifies the location where the slave data will be placed in the master ECOM100 PLC
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed

Parameter	DL05 Range
ECOM100# .....	K
Workspace .....	V
Slave ID .....	K
From Slave Element (Src) X,Y,C,S,T,CT,GX,GY,V,P	X,Y,C,GX,GY,B
Number of Bytes .....	K
To Master Element (Dest) .....	V
Success .....	X,Y,C,GX,GY,B
Error .....	X,Y,C,GX,GY,B

**ECRX Example**

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.



(example continued on next page)

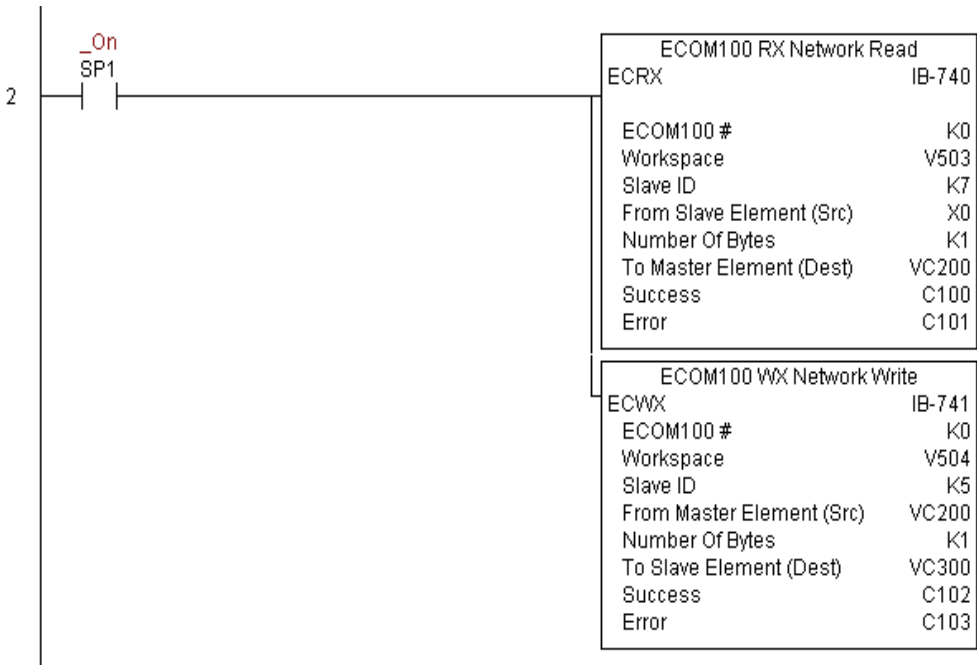
### ECRX Example (cont'd)

Rung 2: Using ECOM100# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the ECRX and ECWX work with the ECOM100 Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP "busy bits" or "error bits", or what slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the ECRX and ECWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending ECOM100 IBoxes below the ECWX, then the very next scan the ECRX would start its request again.

Using the ECRX and ECWX for all of your ECOM100 network reads and writes is the fastest the PLC can do networking. For local Serial Ports, DCM modules, or the original ECOM modules, use the NETCFG and NETRX/NETWX IBoxes.



**ECOM100 WX Network Write(ECWX) (IB-741)**

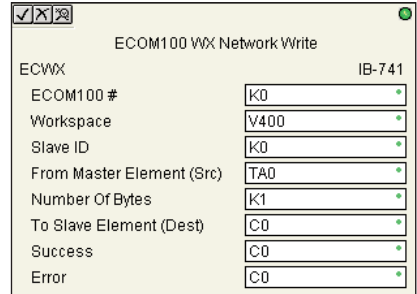
DSS	Used
HPP	N/A

ECOM100 WX Network Write performs the WX instruction with built-in interlocking with all other ECOM100 RX (ECRX) and ECOM100 WX (ECWX) IBoxes in your program to simplify communications networking. It will perform the WX on the specified ECOM100#'s network, which corresponds to a specific unique ECOM100 Configuration (ECOM100) IBox at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Whenever this IBox has power, it will write data from the master's V-memory buffer to the specified slave starting with the given slave element, giving other ECOM100 RX and ECOM100 WX IBoxes on that ECOM100# network a chance to execute.

For example, if you wish to read and write data continuously from 5 different slaves, you can have all of these ECRX and ECWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically.



**ECWX Parameters**

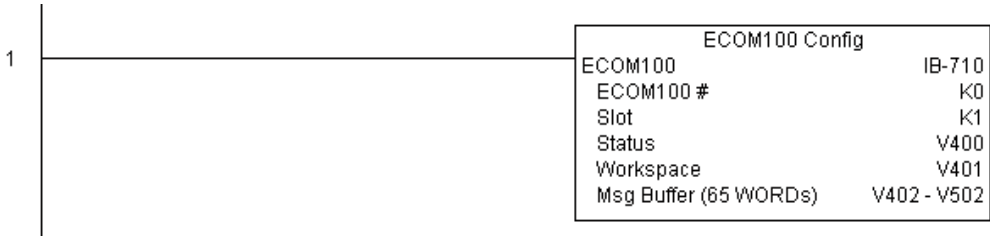
- ECOM100#: this is a logical number associated with this specific ECOM100 module in the specified slot. All other ECxxxx IBoxes that need to reference this ECOM100 module must reference this logical number
- Workspace: specifies a V-memory location that will be used by the instruction
- Slave ID: specifies the slave ECOM(100) PLC that will be targeted by the ECWX instruction
- From Master Element (Src): specifies the location in the master ECOM100 PLC where the data will be sourced from
- Number of Bytes: specifies the number of bytes to write to the slave ECOM(100) PLC
- To Slave Element (Dest): specifies the slave address the data will be written to
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed

Parameter	DL05 Range
ECOM100# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Slave ID ..... K	K0-90
From Master Element (Src) ..... V	See DL05 V-memory map - Data Words
Number of Bytes ..... K	K1-128
To Slave Element (Dest) X,Y,C,S,T,CT,GX,GY,V,P	See DL05 V-memory map
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map

### ECWX Example

Rung 1: The ECOM100 Config IBox is responsible for coordination/interlocking of all ECOM100 type IBoxes for one specific ECOM100 module. Tag the ECOM100 in slot 1 as ECOM100# K0. All other ECxxxx IBoxes refer to this module # as K0. If you need to move the module in the base to a different slot, then you only need to change this one IBox. V400 is used as a global result status register for the other ECxxxx IBoxes using this specific ECOM100 module. V401 is used to coordinate/interlock the logic in all of the other ECxxxx IBoxes using this specific ECOM100 module. V402-V502 is a common 130 byte buffer available for use by the other ECxxxx IBoxes using this specific ECOM100 module.

5





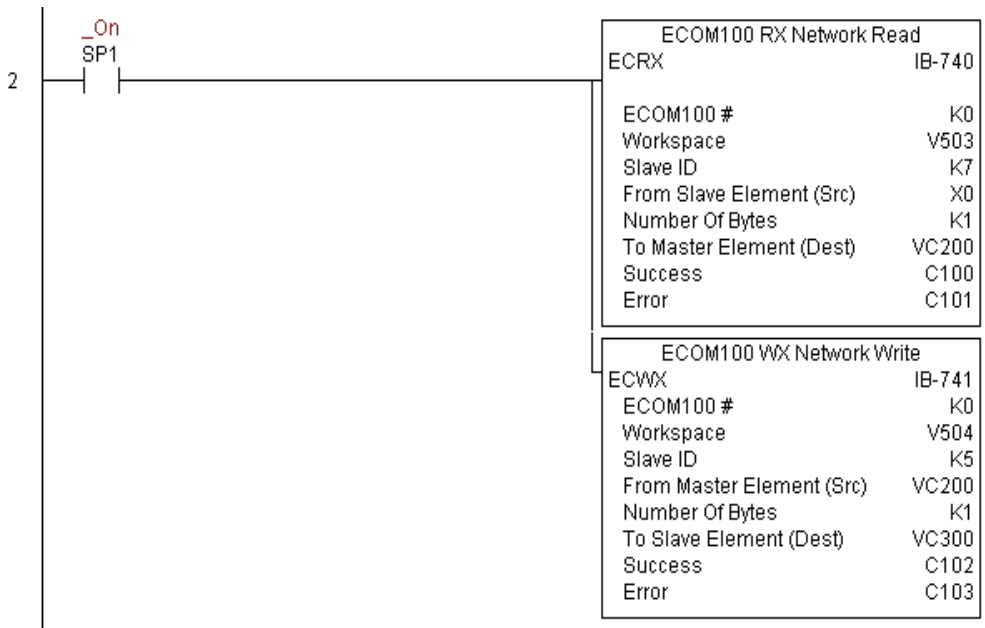
### ECWX Example

Rung 2: Using ECOM100# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the ECRX and ECWX work with the ECOM100 Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP "busy bits" or "error bits", or what slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the ECRX and ECWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending ECOM100 IBoxes below the ECWX, then the very next scan the ECRX would start its request again.

Using the ECRX and ECWX for all of your ECOM100 network reads and writes is the fastest the PLC can do networking. For local Serial Ports, DCM modules, or the original ECOM modules, use the NETCFG and NETRX/NETWX IBoxes.



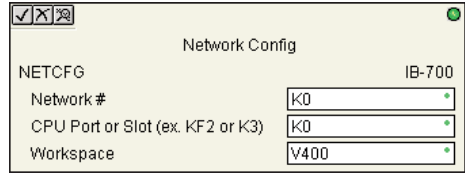
**NETCFG Network Configuration (NETCFG) (IB-700)**

DS5	Used
HPP	N/A

Network Config defines all the common information necessary for performing RX/WX Networking using the NETRX and NETWX IBox instructions via a local CPU serial port, DCM or ECOM module.

You must have the Network Config instruction at the top of your ladder/stage program with any other configuration IBoxes.

If you use more than one local serial port, DCM or ECOM in your PLC for RX/WX Networking, you must have a different Network Config instruction for EACH RX/WX network in your system that utilizes any NETRX/NETWX IBox instructions.



The Workspace parameter is an internal, private register used by the Network Config IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

The 2nd parameter "CPU Port or Slot" is the same value as in the high byte of the first LD instruction if you were coding the RX or WX rung yourself. This value is CPU and port specific (check your PLC manual). Use KF2 for the DL05 CPU serial port 2. If using a DCM or ECOM module, use K1 for slot 1.

**NETCFG Parameters**

- Network#: specifies a unique # for each ECOM(100) or DCM network to use
- CPU Port or Slot: specifies the CPU port number or slot number of DCM/ECOM(100) used
- Workspace: specifies a V-memory location that will be used by the instruction

Parameter	DL05 Range
Network# ..... K	K0-255
CPU Port or Slot ..... K	K0-FF
Workspace ..... V	See DL05 V-memory map - Data Words

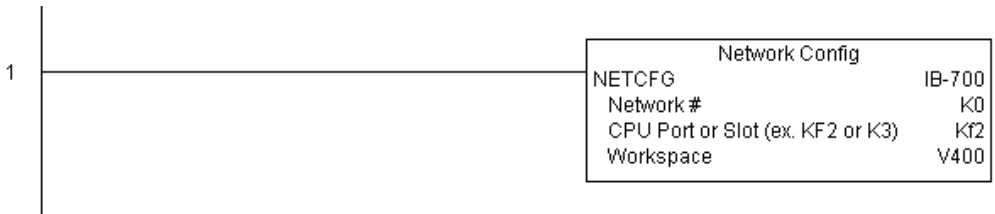
### NETCFG Example

The Network Configuration IBox coordinates all of the interaction with other Network IBoxes (NETRX/NETWX). You must have a Network Configuration IBox for each serial port network, DCM module network, or original ECOM module network in your system. Configuration IBoxes must be at the top of your program and must execute every scan.

This IBox defines Network# K0 to be for the local CPU serial port #2 (KF2). For local CPU serial ports or DCM/ECOM modules, use the same value you would use in the most significant byte of the first LD instruction in a normal RX/WX rung to reference the port or module. Any NETRX or NETWX IBoxes that need to reference this specific network would enter K0 for their Network# parameter.

The Workspace register is used to maintain state information about the port or module, along with proper sharing and interlocking with the other NETRX and NETWX IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.

5



### Network RX Read (NETRX) (IB-701)

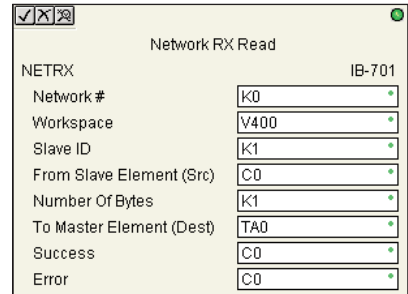
DS5	Used
HPP	N/A

Network RX Read performs the RX instruction with built-in interlocking with all other Network RX (NETRX) and Network WX (NETWX) IBoxes in your program to simplify communications networking. It will perform the RX on the specified Network #, which corresponds to a specific unique Network Configuration (NETCFG) at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Whenever this IBox has power, it will read element data from the specified slave into the given destination V-memory buffer, giving other Network RX and Network WX IBoxes on that Network # a chance to execute.

For example, if you wish to read and write data continuously from 5 different slaves, you can have all of these NETRX and NETWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically.



#### NETRX Parameters

- Network#: specifies the (CPU port's, DCM's, ECOM's) Network # defined by the NETCFG instruction
- Workspace: specifies a V-memory location that will be used by the instruction
- Slave ID: specifies the slave PLC that will be targeted by the NETRX instruction
- From Slave Element (Src): specifies the slave address of the data to be read
- Number of Bytes: specifies the number of bytes to read from the slave device
- To Master Element (Dest): specifies the location where the slave data will be placed in the master PLC
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed

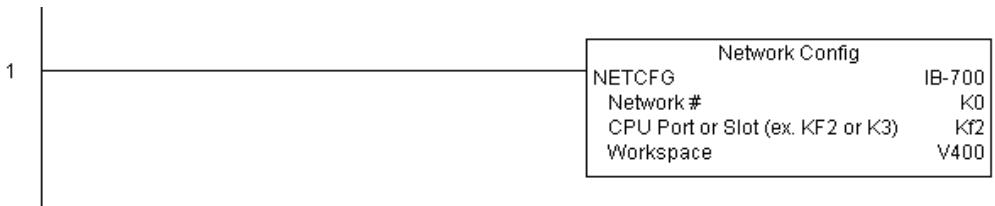
Parameter	DL05 Range
Network# . . . . . K	K0-255
Workspace . . . . . V	See DL05 V-memory map - Data Words
Slave ID . . . . . K	K0-90
From Slave Element (Src) X,Y,C,S,T,CT,GX,GY,V,P	See DL05 V-memory map
Number of Bytes . . . . . K	K1-128
To Master Element (Dest) . . . . . V	See DL05 V-memory map - Data Words
Success . . . . . X,Y,C,GX,GY,B	See DL05 V-memory map
Error . . . . . X,Y,C,GX,GY,B	See DL05 V-memory map

### NETRX Example

Rung 1: The Network Configuration IBox coordinates all of the interaction with other Network IBoxes (NETRX/NETWX). You must have a Network Configuration IBox for each serial port network, DCM module network, or original ECOM module network in your system. Configuration IBoxes must be at the top of your program and must execute every scan.

This IBox defines Network# K0 to be for the local CPU serial port #2 (KF2). For local CPU serial ports or DCM/ECOM modules, use the same value you would use in the most significant byte of the first LD instruction in a normal RX/WX rung to reference the port or module. Any NETRX or NETWX IBoxes that need to reference this specific network would enter K0 for their Network# parameter.

The Workspace register is used to maintain state information about the port or module, along with proper sharing and interlocking with the other NETRX and NETWX IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.



(example continued on next page)

### NETRX Example (cont'd)

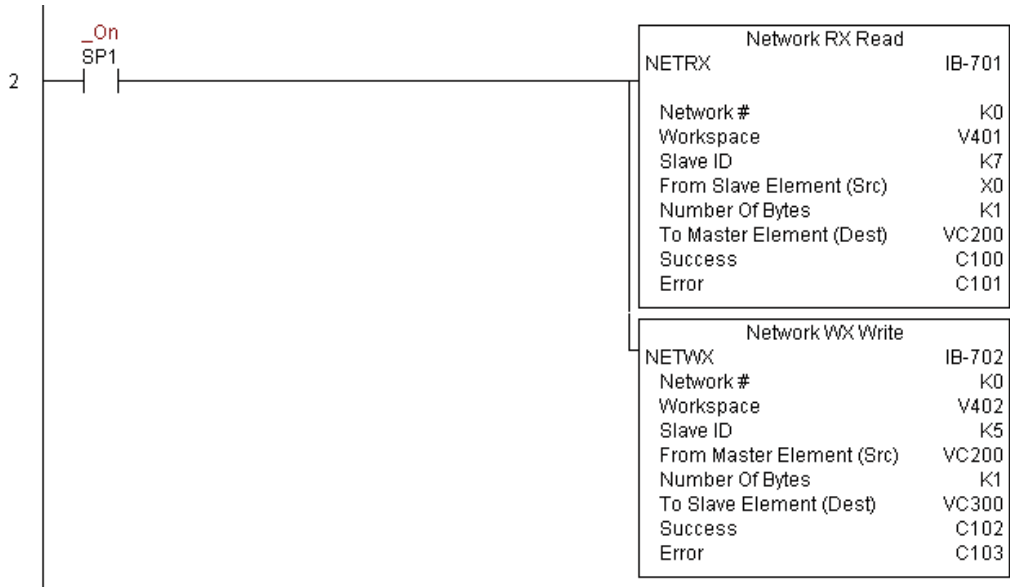
Rung 2: Using Network# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the NETRX and NETWX work with the Network Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP "busy bits" or "error bits", or what port number or slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the NETRX and NETWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending NETRX or NETWX IBoxes below this IBox, then the very next scan the NETRX would start its request again.

Using the NETRX and NETWX for all of your serial port, DCM, or original ECOM network reads and writes is the fastest the PLC can do networking. For ECOM100 modules, use the ECOM100 and ECRX/ECWX IBoxes.

5



### Network WX Write (NETWX) (IB-702)

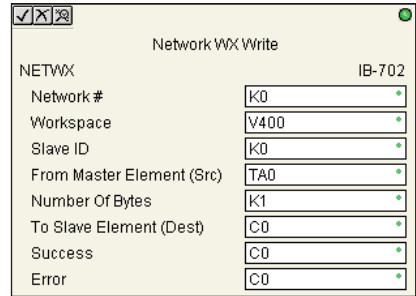
DS5	Used
HPP	N/A

Network WX Write performs the WX instruction with built-in interlocking with all other Network RX (NETRX) and Network WX (NETWX) IBoxes in your program to simplify communications networking. It will perform the WX on the specified Network #, which corresponds to a specific unique Network Configuration (NETCFG) at the top of your program.

The Workspace parameter is an internal, private register used by this IBox and **MUST BE UNIQUE** in this one instruction and **MUST NOT** be used anywhere else in your program.

Whenever this IBox has power, it will write data from the master's V-memory buffer to the specified slave starting with the given slave element, giving other Network RX and Network WX IBoxes on that Network # a chance to execute.

For example, if you wish to read and write data continuously from 5 different slaves, you can have all of these NETRX and NETWX instructions in ONE RUNG driven by SP1 (Always On). They will execute round-robin style, automatically.



#### NETWX Parameters

- Network#: specifies the (CPU port's, DCM's, ECOM's) Network # defined by the NETCFG instruction
- Workspace: specifies a V-memory location that will be used by the instruction
- Slave ID: specifies the slave PLC that will be targeted by the NETWX instruction
- From Master Element (Src): specifies the location in the master PLC where the data will be sourced from
- Number of Bytes: specifies the number of bytes to write to the slave PLC
- To Slave Element (Dest): specifies the slave address the data will be written to
- Success: specifies a bit that will turn on once the request is completed successfully
- Error: specifies a bit that will turn on if the instruction is not successfully completed

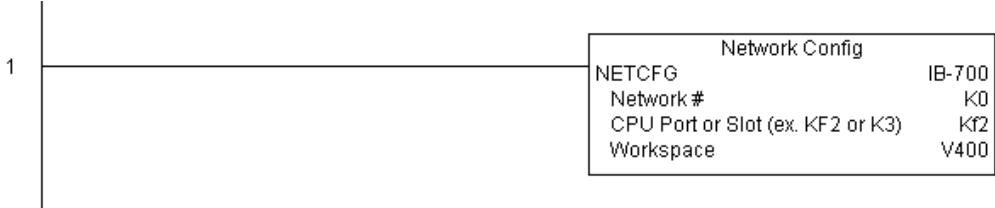
Parameter	DL05 Range
Network# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Slave ID ..... K	K0-90
From Master Element (Src) ..... V	See DL05 V-memory map - Data Words
Number of Bytes ..... K	K1-128
To Slave Element (Dest) X,Y,C,S,T,CT,GX,GY,V,P	See DL05 V-memory map
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map

### NETWX Example

Rung 1: The Network Configuration IBox coordinates all of the interaction with other Network IBoxes (NETRX/NETWX). You must have a Network Configuration IBox for each serial port network, DCM module network, or original ECOM module network in your system. Configuration IBoxes must be at the top of your program and must execute every scan.

This IBox defines Network# K0 to be for the local CPU serial port #2 (KF2). For local CPU serial ports or DCM/ECOM modules, use the same value you would use in the most significant byte of the first LD instruction in a normal RX/WX rung to reference the port or module. Any NETRX or NETWX IBoxes that need to reference this specific network would enter K0 for their Network# parameter.

The Workspace register is used to maintain state information about the port or module, along with proper sharing and interlocking with the other NETRX and NETWX IBoxes in the program. This V-memory register must not be used anywhere else in the entire program.





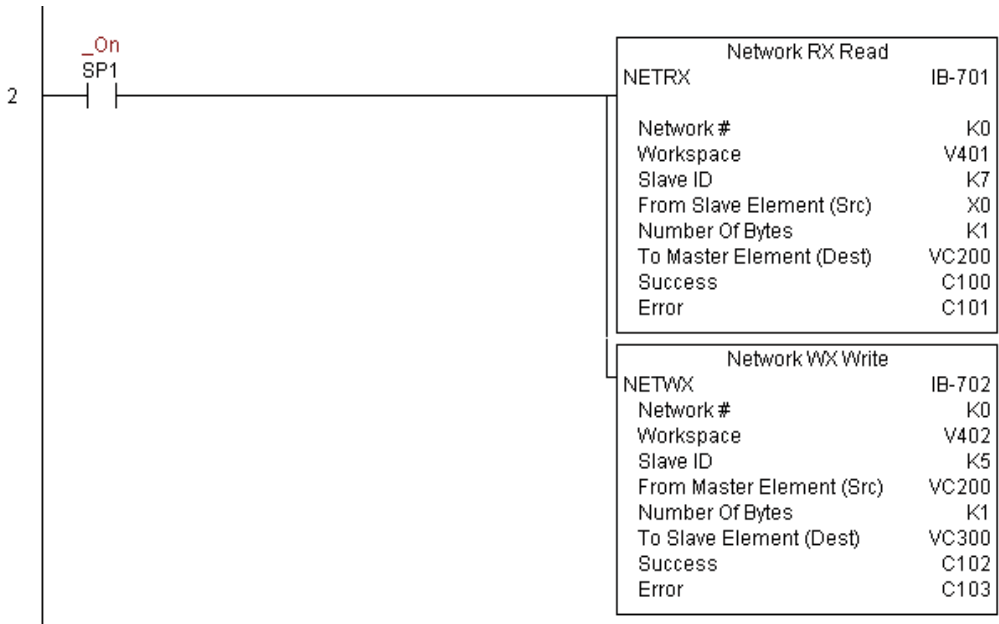
### NETWX Example

Rung 2: Using Network# K0, read X0-X7 from Slave K7 and write them to slave K5 as fast as possible. Store them in this local PLC in C200-C207, and write them to C300-C307 in slave K5.

Both the NETRX and NETWX work with the Network Config IBox to simplify all networking by handling all of the interlocks and proper resource sharing. They also provide very simplified error reporting. You no longer need to worry about any SP "busy bits" or "error bits", or what port number or slot number a module is in, or have any counters or shift registers or any other interlocks for resource management.

In this example, SP1 (always ON) is driving both the NETRX and NETWX IBoxes in the same rung. On the scan that the Network Read completes, the Network Write will start that same scan. As soon as the Network Write completes, any pending operations below it in the program would get a turn. If there are no pending NETRX or NETWX IBoxes below this IBox, then the very next scan the NETRX would start its request again.

Using the NETRX and NETWX for all of your serial port, DCM, or original ECOM network reads and writes is the fastest the PLC can do networking. For ECOM100 modules, use the ECOM100 and ECRX/ECWX IBoxes.

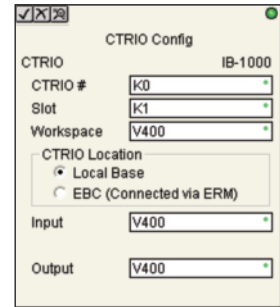


### CTRIO Configuration (CTRIO) (IB-1000)

DS5	Used
HPP	N/A

CTRIO Config defines all the common information for one specific CTRIO module which is used by the other CTRIO IBox instructions (for example, CTRLDPR - CTRIO Load Profile, CTREDRL - CTRIO Edit and Reload Preset Table, CTRRTL - CTRIO Run to Limit Mode, ...).

The Input/Output parameters for this instruction can be copied directly from the CTRIO Workbench configuration for this CTRIO module. Since the behavior is slightly different when the CTRIO module is in an EBC Base via an ERM, you must specify whether the CTRIO module is in a local base or in an EBC base. The DL05 PLC only supports local base operation at this time.



You must have the CTRIO Config IBox at the top of your ladder/stage program along with any other configuration IBoxes.

If you have more than one CTRIO in your PLC, you must have a different CTRIO Config IBox for EACH CTRIO module in your system that utilizes any CTRIO IBox instructions. Each CTRIO Config IBox must have a UNIQUE CTRIO# value. This is how the CTRIO IBoxes differentiate between the different CTRIO modules in your system.

The Workspace parameter is an internal, private register used by the CTRIO Config IBox and MUST BE UNIQUE in this one instruction and MUST NOT be used anywhere else in your program.

#### CTRIO Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number
- Slot: specifies the single PLC option slot the CTRIO module occupies
- Workspace: specifies a V-memory location that will be used by the instruction
- CTRIO Location: specifies where the module is located (local base only for DL05)
- Input: This needs to be set to the same V-memory register as is specified in CTRIO Workbench as 'Starting V address for inputs' for this unique CTRIO.
- Output: This needs to be set to the same V-memory register as is specified in CTRIO Workbench as 'Starting V address for outputs' for this unique CTRIO.

Parameter	DL05 Range
CTRIO# ..... K	K0-255
Slot ..... K	K1
Workspace ..... V	See DL05 V-memory map - Data Words
Input ..... V	See DL05 V-memory map - Data Words
Output ..... V	See DL05 V-memory map - Data Words

### CTRIO Example

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



### CTRIO Add Entry to End of Preset Table (CTRADPT) (IB-1005)

DS5	Used
HPP	N/A

CTRIO Add Entry to End of Preset Table, on a leading edge transition to this IBox, will append an entry to the end of a memory based Preset Table on a specific CTRIO Output resource. This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

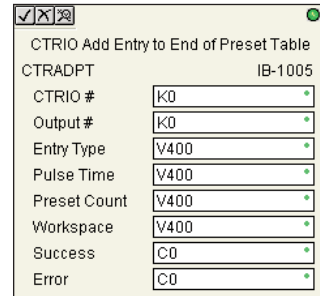
K3: Pulse Off (uses Pulse Time)

K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.



#### CTRAPT Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config)
- Output#: specifies a CTRIO output to be used by the instruction
- Entry Type: specifies the Entry Type to be added to the end of a Preset Table
- Pulse Time: specifies a pulse time for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter	DL05 Range
CTRIO# ..... K	K0-255
Output# ..... K	K0-3
Entry Type ..... V,K	K0-5; See DL05 V-memory map - Data Words
Pulse Time ..... V,K	K0-65535; See DL05 V-memory map - Data Words
Preset Count ..... V,K	K0-2147434528; See DL05 V-memory map
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map

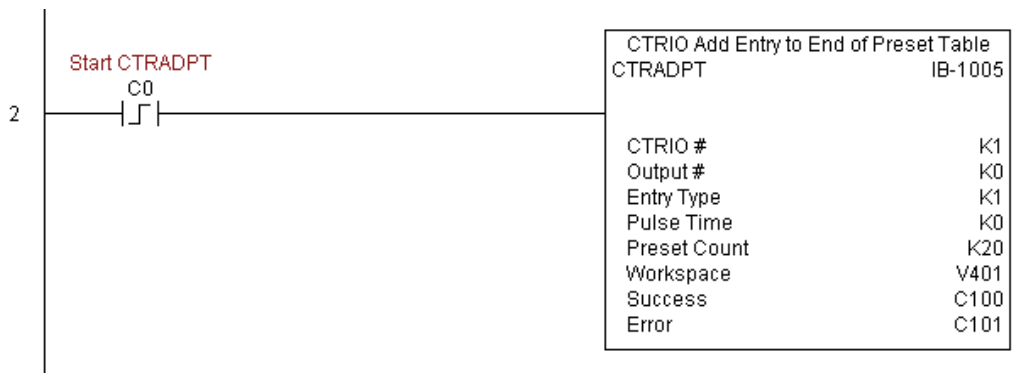
## CTRADPT Example

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



Rung 2: This rung is a sample method for enabling the CTRADPT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes. Turning on C0 will cause the CTRADPT instruction to add a new preset to the preset table for output #0 on the CTRIO in slot 2. The new preset will be a command to RESET (entry type K1=reset), pulse time is left at zero as the reset type does not use this, and the count at which it will reset will be 20.

Operating procedure for this example code is to load the CTRADPT\_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in dataview, turn encoder on CTRIO to value above 10 and output #0 light will come on and stay on for all counts past 10. Now reset the counter with C1, enable C0 to execute CTRADPT command to add a reset for output #0 at a count of 20, turn on C2 to enable output #0, then turn encoder to value of 10+ (output #0 should turn on) and then continue on to count of 20+ (output #0 should turn off).



(example continued on next page)

### CTRADPT Example (cont'd)

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.

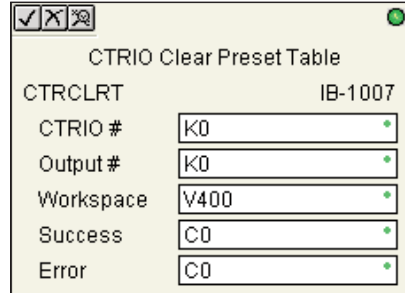


**CTRIO Clear Preset Table (CTRCLRT) (IB-1007)**

DS5	Used
HPP	N/A

CTRIO Clear Preset Table will clear the RAM based Preset Table on a leading edge transition to this IBox. This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.



**CTRCLRT Parameters**

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config)
- Output#: specifies a CTRIO output to be used by the instruction
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter	DL05 Range
CTRIO# .....	K
Output# .....	K
Workspace .....	V
Success .....	X,Y,C,GX,GY,B
Error .....	X,Y,C,GX,GY,B
	K0-255
	K0-3
	See DL05 V-memory map - Data Words
	See DL05 V-memory map
	See DL05 V-memory map

### CTRCLRT Example

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



Rung 2: This rung is a sample method for enabling the CTRCLRT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes. Turning on C0 will cause the CTRCLRT instruction to clear the preset table for output #0 on the CTRIO in slot 2.

Operating procedure for this example code is to load the CTRCLRT\_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on and stay on until a count of 20 is reached, where it will turn off. Now reset the counter with C1, enable C0 to execute CTRCLRT command to clear the preset table, turn on C2 to enable output #0, then turn encoder to value of 10+ (output #0 should NOT turn on).



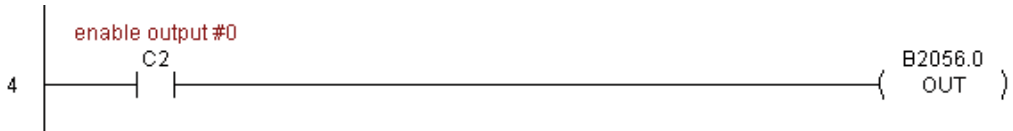


**CTRCLRT Example**

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



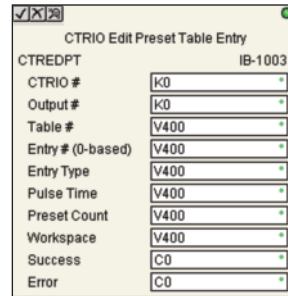
Rung 4: This rung allows the operator to enable output #0 from the ladder code.



### CTRIO Edit Preset Table Entry (CTREDPT) (IB-1003)

DS5	Used
HPP	N/A

CTRIO Edit Preset Table Entry, on a leading edge transition to this IBox, will edit a single entry in a Preset Table on a specific CTRIO Output resource. This IBox is good if you are editing more than one entry in a file at a time. If you wish to do just one edit and then reload the table immediately, see the CTRIO Edit and Reload Preset Table Entry (CTREDRL) IBox. This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.



Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

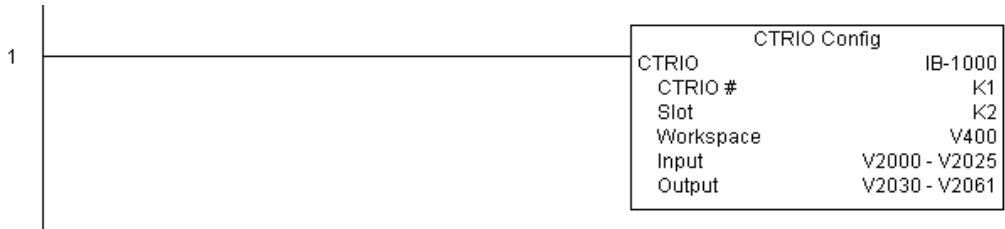
#### CTREDPT Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Table#: specifies the Table number of which an Entry is to be edited
- Entry#: specifies the Entry location in the Preset Table to be edited
- Entry Type: specifies the Entry Type to add during the edit
- Pulse Time: specifies a pulse time for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter	DL05 Range
CTRIO# ..... K	K0-255
Output# ..... K	K0-3
Table# ..... V,K	K0-255; See DL05 V-memory map - Data Words
Entry# ..... V,K	K0-255; See DL05 V-memory map - Data Words
Entry Type ..... V,K	K0-5; See DL05 V-memory map - Data Words
Pulse Time ..... V,K	K0-65535; See DL05 V-memory map - Data Words
Preset Count ..... V,K	K0-2147434528; See DL05 V-memory map
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map

**CTREDPT Example**

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



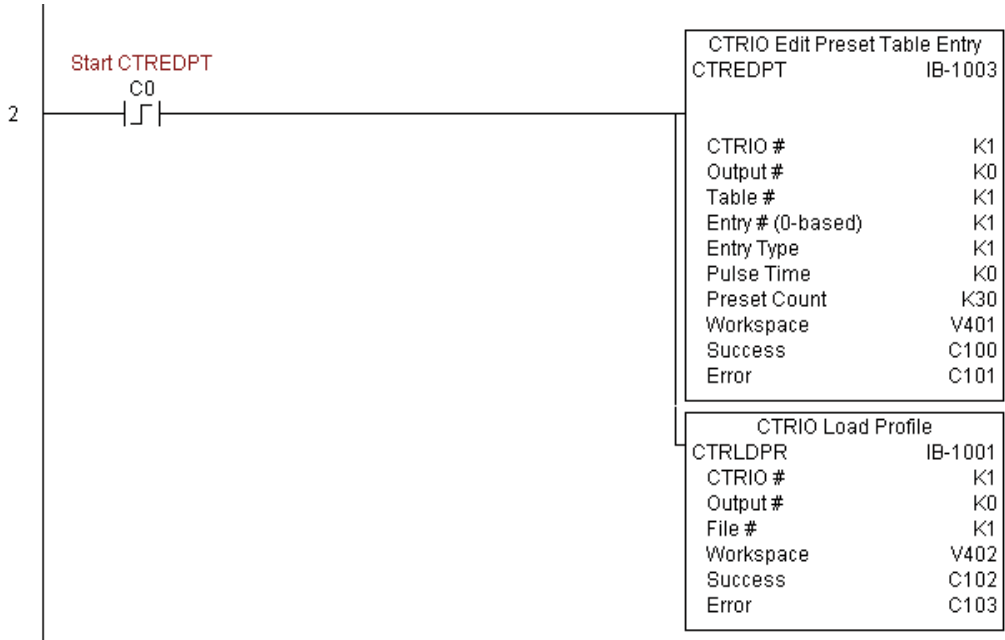
(example continued on next page)

### CTREDPT Example (cont'd)

Rung 2: This rung is a sample method for enabling the CTREDPT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes. Turning on C0 will cause the CTREDPT instruction to change the second preset from a reset at a count of 20 to a reset at a count of 30 for output #0 on the CTRIO in slot 2.

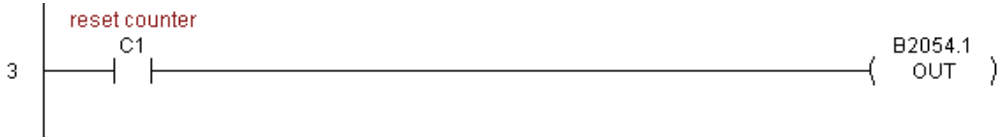
Operating procedure for this example code is to load the CTREDPT\_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on and stay on until a count of 20 is reached, where it will turn off. Now reset the counter with C1, enable C0 to execute CTREDPT command to change the second preset, turn on C2 to enable output #0, then turn encoder to value of 10+ (output #0 should turn on) and then continue past a count of 30 (output #0 should turn off).

Note that we must also reload the profile after changing the preset(s), this is why the CTRLDPR command follows the CTREDPT command in this example.



**CTREDPT Example**

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.



## CTRIO Edit Preset Table Entry and Reload (CTREDRL) (IB-1002)

DS5	Used
HPP	N/A

CTRIO Edit Preset Table Entry and Reload, on a leading edge transition to this IBox, will perform this dual operation to a CTRIO Output resource in one CTRIO command. This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and MUST NOT be used anywhere else in your program.

### CTREDRL Parameters

- **CTRIO#:** specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- **Output#:** specifies a CTRIO output to be used by the instruction
- **Table#:** specifies the Table number of which an Entry is to be edited
- **Entry#:** specifies the Entry location in the Preset Table to be edited
- **Entry Type:** specifies the Entry Type to add during the edit
- **Pulse Time:** specifies a pulse time for the Pulse On and Pulse Off Entry Types
- **Preset Count:** specifies an initial count value to begin at after Reset
- **Workspace:** specifies a V-memory location that will be used by the instruction
- **Success:** specifies a bit that will turn on once the instruction has successfully completed
- **Error:** specifies a bit that will turn on if the instruction does not complete successfully

Parameter	DL05 Range
CTRIO# ..... K	K0-255
Output# ..... K	K0-3
Table# ..... V,K	K0-255; See DL05 V-memory map - Data Words
Entry# ..... V,K	K0-255; See DL05 V-memory map - Data Words
Entry Type ..... V,K	K0-5; See DL05 V-memory map - Data Words
Pulse Time ..... V,K	K0-65535; See DL05 V-memory map - Data Words
Preset Count ..... V,K	K0-2147434528; See DL05 V-memory map
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map

### CTREDRL Example

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



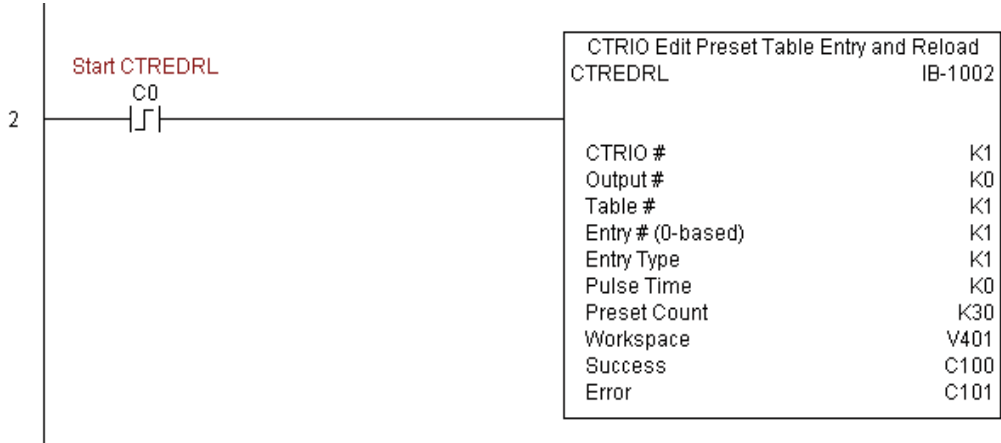
(example continued on next page)

**CTREDRL Example (cont'd)**

Rung 2: This rung is a sample method for enabling the CTREDRL command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes. Turning on C0 will cause the CTREDRL instruction to change the second preset in file 1 from a reset at a value of 20 to a reset at a value of 30.

Operating procedure for this example code is to load the CTREDRL\_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on, continue to a count above 20 and the output #0 light will turn off. Now reset the counter with C1, enable C0 to execute CTREDRL command to change the second preset count value to 30, then turn encoder to value of 10+ (output #0 should turn on) and continue on to a value of 30+ and the output #0 light will turn off.

Note that it is not necessary to reload this file separately, however, the command can only change one value at a time.





## CTREDRL Example

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.



### CTRIO Initialize Preset Table (CTRINPT) (IB-1004)

DS5	Used
HPP	N/A

CTRIO Initialize Preset Table, on a leading edge transition to this IBox, will create a single entry Preset Table in memory but not as a file, on a specific CTRIO Output resource. This IBox will take more than 1 PLC scan to execute.

Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

#### CTRINPT Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Entry Type: specifies the Entry Type to add during the edit
- Pulse Time: specifies a pulse time for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter	DL05 Range
CTRIO# ..... K	K0-255
Output# ..... K	K0-3
Entry Type ..... V,K	K0-5; See DL05 V-memory map - Data Words
Pulse Time ..... V,K	K0-65535; See DL05 V-memory map - Data Words
Preset Count ..... V,K	K0-2147434528; See DL05 V-memory map
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map

### CTRINPT Example

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



(example continued on next page)

### CTRINPT Example (cont'd)

Rung 2: This rung is a sample method for enabling the CTRINPT command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTRINPT instruction to create a single entry preset table, but not as a file, and use it for the output #0. In this case the single preset will be a set at a count of 15 for output #0.

Operating procedure for this example code is to load the CTRINPT\_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 15 and output #0 light will not come on. Now reset the counter with C1, enable C0 to execute CTRINPT command to create a single preset table with a preset to set output#0 at a count of 15, then turn encoder to value of 15+ (output #0 should turn on).

5



### CTRINPT Example

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.



Rung 4: This rung allows the operator to enable output #0 from the ladder code.



## CTRIO Initialize Preset Table (CTRINTR) (IB-1010)

DS5	Used
HPP	N/A

CTRIO Initialize Preset Table, on a leading edge transition to this IBox, will create a single entry Preset Table in memory but not as a file, on a specific CTRIO Output resource. This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

Entry Type:

K0: Set

K1: Reset

K2: Pulse On (uses Pulse Time)

K3: Pulse Off (uses Pulse Time)

K4: Toggle

K5: Reset Count

Note that the Pulse Time parameter is ignored by some Entry Types.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

Parameter	Value
CTRINTR	IB-1010
CTRIO #	K0
Output #	K0
Entry Type	V400
Pulse Time	V400
Preset Count	V400
Workspace	V400
Success	C0
Error	C0

### CTRINTR Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Entry Type: specifies the Entry Type to add during the edit
- Pulse Time: specifies a pulse time for the Pulse On and Pulse Off Entry Types
- Preset Count: specifies an initial count value to begin at after Reset
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter	DL05 Range
CTRIO# ..... K	K0-255
Output# ..... K	K0-3
Entry Type ..... V,K	K0-5; See DL05 V-memory map - Data Words
Pulse Time ..... V,K	K0-65535; See DL05 V-memory map - Data Words
Preset Count ..... V,K	K0-2147434528; See DL05 V-memory map
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map

**CTRINTR Example**

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



(example continued on next page)

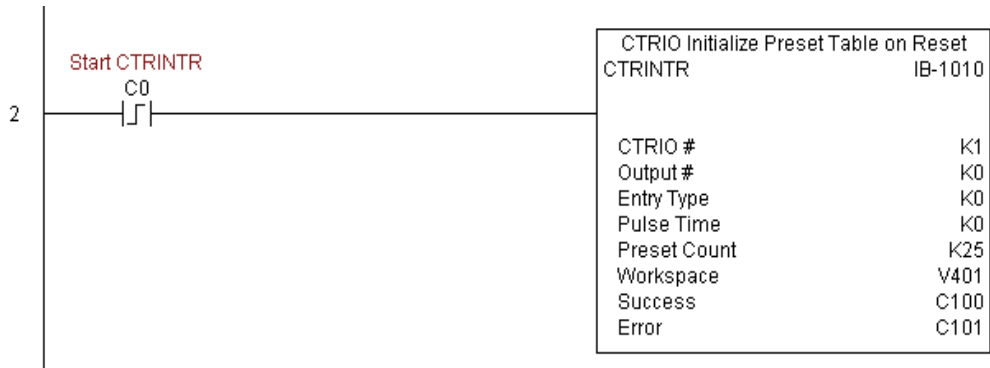
### CTRINTR Example (cont'd)

Rung 2: This rung is a sample method for enabling the CTRINTR command. A C-bit is used to allow the programmer to control the command from Data View for testing purposes.

Turning on C0 will cause the CTRINTR instruction to create a single entry preset table, but not as a file, and use it for output #0, the new preset will be loaded when the current count is reset. In this case the single preset will be a set at a count of 25 for output #0.

Operating procedure for this example code is to load the CTRINTR\_ex1.cwb file to your CTRIO, then enter the code shown here, change to RUN mode, enable output #0 by turning on C2 in Data View, turn encoder on CTRIO to value above 10 and output #0 light will come on. Now turn on C0 to execute the CTRINTR command, reset the counter with C1, then turn encoder to value of 25+ (output #0 should turn on).

5



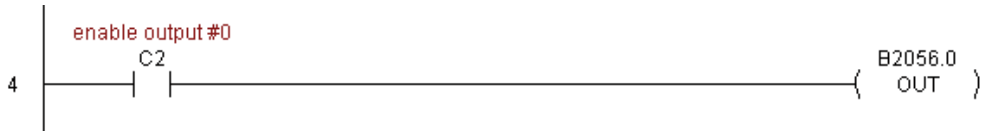


### CTRINTR Example

Rung 3: This rung allows the programmer to reset the counter from the ladder logic.

**5**

Rung 4: This rung allows the operator to enable output #0 from the ladder code.

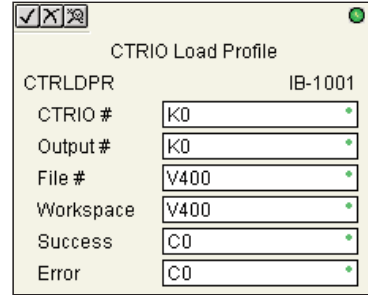


**CTRIO Load Profile (CTRLDPR) (IB-1001)**

DS5	Used
HPP	N/A

CTRIO Load Profile loads a CTRIO Profile File to a CTRIO Output resource on a leading edge transition to this IBox. This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.



**CTRLDPR Parameters**

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config)
- Output#: specifies a CTRIO output to be used by the instruction
- File#: specifies a CTRIO profile File number to be loaded
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter	DL05 Range
CTRIO# ..... K	K0-255
Output# ..... K	K0-3
File# ..... V,K	K0-255; See DL05 V-memory map - Data Words
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map

### CTRLDPR Example

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



Rung 2: This CTRIO Load Profile IBox will load File #1 into the working memory of Output 0 in CTRIO #1. This example program requires that you load CTRLDPR\_IBox.cwb into your Hx-CTRIO(2) module.



Rung 3: If the file is successfully loaded, set Profile\_Loaded.



### CTRIO Read Error (CTRRDER) (IB-1014)

DS5	Used
HPP	N/A

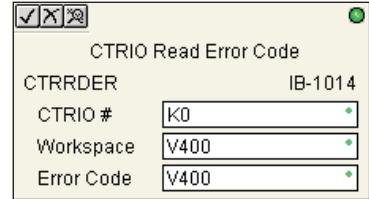
CTRIO Read Error Code will get the decimal error code value from the CTRIO module (listed below) and place it into the given Error Code register, on a leading edge transition to the IBox

Since the Error Code in the CTRIO is only maintained until another CTRIO command is given, you must use this instruction immediately after the CTRIO IBox that reports an error via its Error bit parameter.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

Error Codes:

- 0: No Error
- 100: Specified command code is unknown or unsupported
- 101: File number not found in the file system
- 102: File type is incorrect for specified output function
- 103: Profile type is unknown
- 104: Specified input is not configured as a limit on this output
- 105: Specified limit input edge is out of range
- 106: Specified input function is unconfigured or invalid
- 107: Specified input function number is out of range
- 108: Specified preset function is invalid
- 109: Preset table is full
- 110: Specified Table entry is out of range
- 111: Specified register number is out of range
- 112: Specified register is an unconfigured input or output
- 2001: Error reading Error Code - cannot access CTRIO via ERM



#### CTRRDER Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config)
- Workspace: specifies a V-memory location that will be used by the instruction
- Error Code: specifies the location where the Error Code will be written

Parameter	DL05 Range
CTRIO# ..... K	K0-255
Workspace ..... V	See DL05 V-memory map - Data Words
Error Code ..... V	See DL05 V-memory map - Data Words

### CTRRDER Example

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



5

Rung 2: This CTRIO Read Error Code IBox will read the Extended Error information from CTRIO #1. This example program requires that you load CTRRDER\_IBox.cwb into your Hx-CTRIO(2) module.



### CTRIO Run to Limit Mode (CTRRTLM) (IB-1011)

DS5	Used
HPP	N/A

CTRIO Run To Limit Mode, on a leading edge transition to this IBox, loads the Run to Limit command and given parameters on a specific Output resource. The CTRIO's Input(s) must be configured as Limit(s) for this function to work.

Valid Hexadecimal Limit Values:

K00 - Rising Edge of Ch1/C

K10 - Falling Edge of Ch1/C

K20 - Both Edges of Ch1/C

K01 - Rising Edge of Ch1/D

K11 - Falling Edge of Ch1/D

K21 - Both Edges of Ch1/D

K02 - Rising Edge of Ch2/C

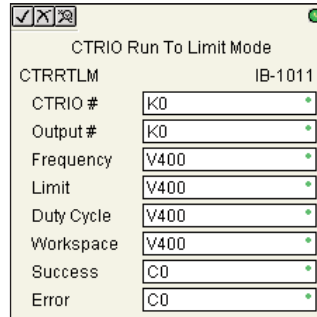
K12 - Falling Edge of Ch2/C

K22 - Both Edges of Ch2/C

K03 - Rising Edge of Ch2/D

K13 - Falling Edge of Ch2/D

K23 - Both Edges of Ch2/D



This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

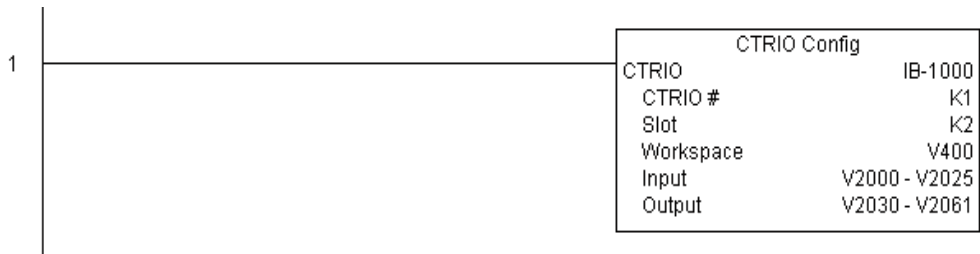
#### CTRRTLM Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Frequency: specifies the output pulse rate (H0-CTRIO: 20Hz - 25KHz / H0-CTRIO2: 20Hz - 250 KHz)
- Limit: the CTRIO's Input(s) must be configured as Limit(s) for this function to operate
- Duty Cycle: specifies the % of on time versus off time. This is a hex number. Default of 0 is 50%, also entering 50 will yield 50%. 50% duty cycle is defined as on half the time and off half the time
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

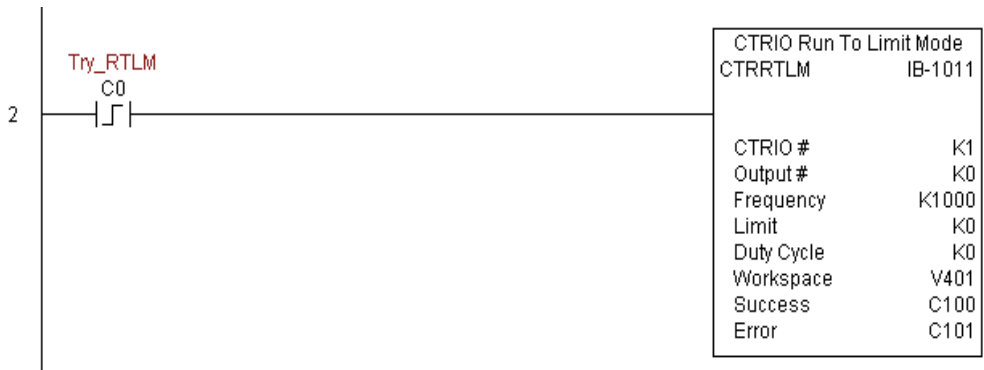
Parameter	DL05 Range
CTRIO# ..... K	K0-255
Output# ..... K	K0-3
Frequency ..... V,K	K20-20000; See DL05 V-memory map - Data Words
Limit ..... V,K	K0-FF; See DL05 V-memory map - Data Words
Duty Cycle ..... V,K	K0-99; See DL05 V-memory map - Data Words
Workspace ..... V	See DL05 V-memory map - Data Words
Success ..... X,Y,C,GX,GY,B	See DL05 V-memory map
Error ..... X,Y,C,GX,GY,B	See DL05 V-memory map

### CTRRTLM Example

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



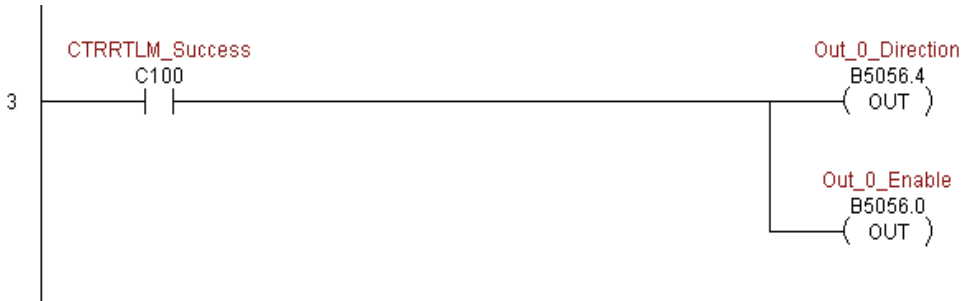
Rung 2: This CTRIO Run To Limit Mode IBox sets up Output #0 in CTRIO #1 to output pulses at a Frequency of 1000 Hz until Llimit #0 comes on. This example program requires that you load CTRRTLM\_IBox.cwb into your Hx-CTRIO(2) module.



(example continued on next page)

### CTRRTLM Example (cont'd)

Rung 3: If the Run To Limit Mode parameters are OK, set the Direction Bit and Enable the output.





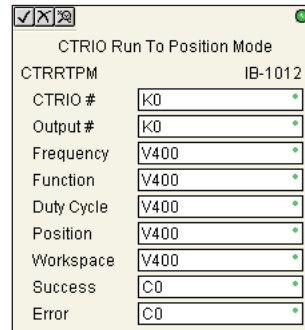
### CTRIO Run to Position Mode (CTRRTPM) (IB-1012)

CTRIO Run To Position Mode, on a leading edge transition to this IBox, loads the Run to Position command and given parameters on a specific Output resource.

DS5	Used
HPP	N/A

Valid Function Values are:

- 00: Less Than Ch1/Fn1
- 10: Greater Than Ch1/Fn1
- 01: Less Than Ch1/Fn2
- 11: Greater Than Ch1/Fn2
- 02: Less Than Ch2/Fn1
- 12: Greater Than Ch2/Fn1
- 03: Less Than Ch2/Fn2
- 13: Greater Than Ch2/Fn2



This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRORDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.

#### CTRRTPM Parameters

- CTRIO#: specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- Output#: specifies a CTRIO output to be used by the instruction
- Frequency: specifies the output pulse rate (H0-CTRIO: 20Hz - 25KHz / H0-CTRIO2: 20Hz - 250 KHz)
- Duty Cycle: specifies the % of on time versus off time. This is a hex number. Default of 0 is 50%, also entering 50 will yield 50%. 50% duty cycle is defined as on half the time and off half the time
- Position: specifies the count value, as measured on the encoder input, at which the output pulse train will be turned off
- Workspace: specifies a V-memory location that will be used by the instruction
- Success: specifies a bit that will turn on once the instruction has successfully completed
- Error: specifies a bit that will turn on if the instruction does not complete successfully

Parameter	DL05 Range
CTRIO# . . . . . K	K0-255
Output# . . . . . K	K0-3
Frequency . . . . . V,K	K20-20000; See DL05 V-memory map - Data Words
Duty Cycle . . . . . V,K	K0-99; See DL05 V-memory map
Position . . . . . V,K	K0-2147434528; See DL05 V-memory map
Workspace . . . . . V	See DL05 V-memory map - Data Words
Success . . . . . X,Y,C,GX,GY,B	See DL05 V-memory map
Error . . . . . X,Y,C,GX,GY,B	See DL05 V-memory map

### CTRRTPM Example

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



Rung 2: This CTRIO Run To Position Mode IBox sets up Output #0 in CTRIO #1 to output pulses at a Frequency of 1000 Hz, use the 'Greater than Ch1/Fn1' comparison operator, until the input position of 1500 is reached. This example program requires that you load CTRRTPM\_IBox.cwb into your Hx-CTRIO(2) module.



Rung 3: If the Run To Position Mode parameters are OK, set the Direction Bit and Enable the output.



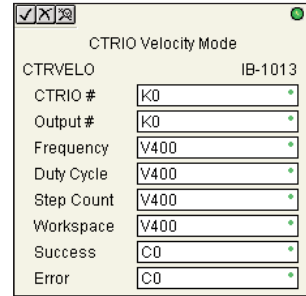
### CTRIO Velocity Mode (CTRVELO) (IB-1013)

DS5	Used
HPP	N/A

CTRIO Velocity Mode loads the Velocity command and given parameters on a specific Output resource on a leading edge transition to this IBox.

This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.



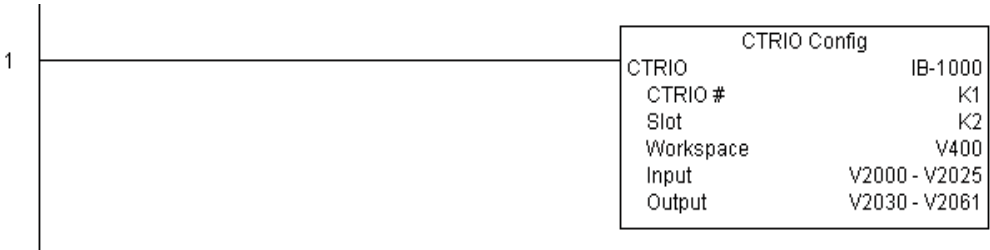
#### CTRVELO Parameters

- **CTRIO#:** specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- **Output#:** specifies a CTRIO output to be used by the instruction
- **Frequency:** specifies the output pulse rate (H0-CTRIO: 20Hz - 25KHz / H0-CTRIO2: 20Hz - 250 KHz)
- **Duty Cycle:** specifies the % of on time versus off time. This is a hex number. Default of 0 is 50%, also entering 50 will yield 50%. 50% duty cycle is defined as on half the time and off half the time
- **Step Count:** specifies the target position as a 32-bit Hex number, a value of Kffffff will cause the profile to run continuously as long as the output is enabled
- **Workspace:** specifies a V-memory location that will be used by the instruction
- **Success:** specifies a bit that will turn on once the instruction has successfully completed
- **Error:** specifies a bit that will turn on if the instruction does not complete successfully

Parameter	DL05 Range
CTRIO# . . . . . K	K0-255
Output# . . . . . K	K0-3
Frequency . . . . . V,K	K20-20000; See DL05 V-memory map - Data Words
Duty Cycle . . . . . V,K	K0-99; See DL05 V-memory map
Step Count . . . . . V,K	K0-2147434528; See DL05 V-memory map
Workspace . . . . . V	See DL05 V-memory map - Data Words
Success . . . . . X,Y,C,GX,GY,B	See DL05 V-memory map
Error . . . . . X,Y,C,GX,GY,B	See DL05 V-memory map

### CTRVELO Example

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



Rung 2: This CTRIO Velocity Mode IBox sets up Output #0 in CTRIO #1 to output 10,000 pulses at a Frequency of 1000 Hz. This example program requires that you load CTRVELO\_IBox.cwb into your Hx-CTRIO(2) module.



### CTRVELO Example

Rung 3: If the Velocity Mode parameters are OK, set the Direction Bit and Enable the output.

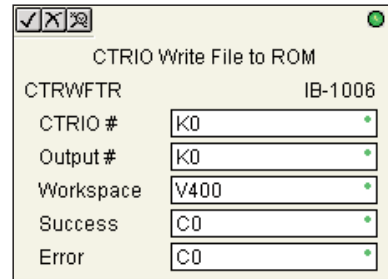
**5**

### CTRIO Write File to ROM (CTRWFTTR) (IB-1006)

DS5	Used
HPP	N/A

CTRIO Write File to ROM writes the runtime changes made to a loaded CTRIO Preset Table back to Flash ROM on a leading edge transition to this IBox. This IBox will take more than 1 PLC scan to execute. Either the Success or Error bit will turn on when the command is complete. If the Error Bit is on, you can use the CTRIO Read Error Code (CTRRDER) IBox to get extended error information.

The Workspace register is for internal use by this IBox instruction and **MUST NOT** be used anywhere else in your program.



#### CTRWFTTR Parameters

- **CTRIO#:** specifies a specific CTRIO module based on a user defined number (see CTRIO Config Ibox)
- **Output#:** specifies a CTRIO output to be used by the instruction
- **Workspace:** specifies a V-memory location that will be used by the instruction
- **Success:** specifies a bit that will turn on once the instruction has successfully completed
- **Error:** specifies a bit that will turn on if the instruction does not complete successfully

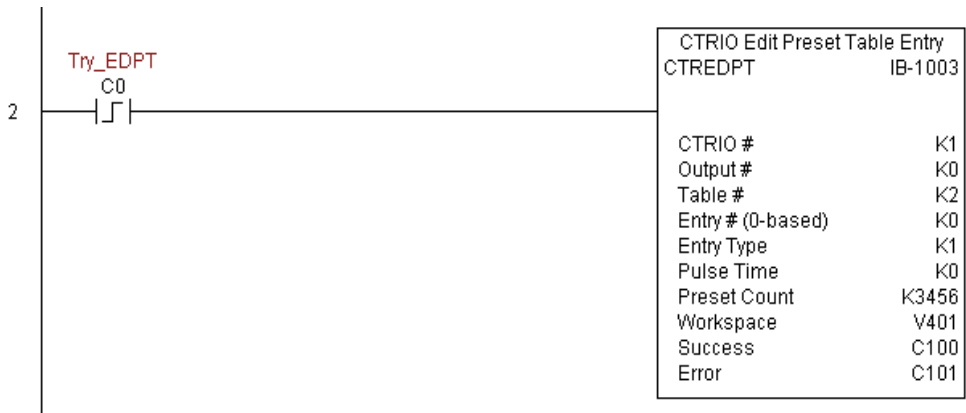
Parameter	DL05 Range	
CTRIO# .....	K	K0-255
Output# .....	K	K0-3
Workspace .....	V	See DL05 V-memory map - Data Words
Success .....	X,Y,C,GX,GY,B	See DL05 V-memory map
Error .....	X,Y,C,GX,GY,B	See DL05 V-memory map

### CTRFTR Example

Rung 1: This sets up the CTRIO card in slot 2 of the local base. Each CTRIO in the system will need a separate CTRIO I-box before any CTRxxxx I-boxes can be used for them. The CTRIO has been configured to use V2000 through V2025 for its input data, and V2030 through V2061 for its output data.



Rung 2: This CTRIO Edit Preset Table Entry IBox will change Entry 0 in Table #2 to be a RESET at Count 3456. This example program requires that you load CTRWFTR\_IBox.cwb into your Hx-CTRIO(2) module.



(example continued on next page)

### CTRWFTR Example (cont'd)

Rung 3: If the file is successfully edited, use a Write File To ROM IBox to save the edited table back to the CTRIO's ROM, thereby making the changes retentive.

