

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

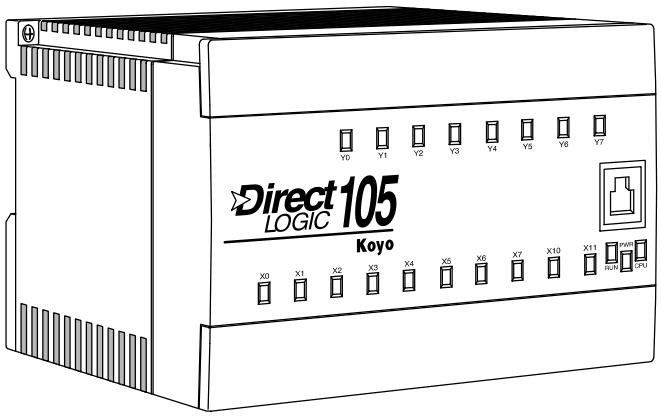
Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

DL105 PLC User Manual

Manual Number D1-USER-M



WARNING

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

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

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

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

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

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

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

Trademarks

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

Copyright 2009, *Automationdirect.com*™ Incorporated
All Rights Reserved

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

AVERTISSEMENT

Nous vous remercions d'avoir acheté l'équipement d'automatisation de **Automationdirect.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 commerc

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 2009, Automationdirect.com Incorporated
Tous droits réservés

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

Manual Revisions

If you contact us in reference to this manual, remember to include the revision number.

Title: DL105 Micro PLC User Manual
Manual Number: D1-USER-M

Edition/Rev	Date	Description of Changes
Original	9/96	Original Issue
2nd Edition	5/98	Updated
2nd Edition, Rev. A	7/03	Updated with corrections and a new appendix
3rd Edition	4/10	Updated with corrections and added Appendix G

Table of Contents

Chapter 1: Getting Started

Introduction **1-2**
The Purpose of this Manual 1-2
Where to Begin 1-2
Supplemental Manuals 1-2
Technical Support 1-2
Conventions Used **1-3**
Key Topics for Each Chapter 1-3
DL105 Micro PLC Components **1-4**
The DL105 Micro PLC Family 1-4
Programming Methods **1-4**
DirectSOFT Programming for Windows™ 1-4
Handheld Programmer 1-5
I/O Quick Chart Selection **1-5**
Quick Start for PLC Checkout and Programming **1-6**
Step 1: Unpack the Recommended DL105 Equipment 1-6
Step 2: Connect Switches to Input Terminals 1-7
Step 3: Connect the Power Wiring 1-8
Step 4: Connect the Programming Device 1-8
Step 5: Switch on the System Power 1-9
Step 6: Initialize Scratchpad Memory 1-9
Step 7: Enter a Ladder Program 1-9
Steps to Designing a Successful System **1-10**
Step 1: Review the Installation Guidelines 1-10
Step 2: Understand the PLC Setup Procedures 1-10
Step 3: Review the I/O Selection Criteria 1-10
Step 4: Choose a System Wiring Strategy 1-10
Step 5: Understand the System Operation 1-10
Step 6: Review the Programming Concepts 1-11
Step 7: Choose the Instructions 1-11
Step 8: Understand the Maintenance and Troubleshooting Procedures 1-11
Questions and Answers about DL105 Micro PLCs **1-12**

Chapter 2: Installation, Wiring, and Specifications

Safety Guidelines **2-2**
Plan for Safety 2-2
Safety Techniques 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
Orientation to DL105 Front Panel 2–5
Accessing the I/O Terminals 2–5
Protective Sheet for DL105 Vents 2–6
Connector Removal 2–6
Mounting Guidelines 2–7
Unit Dimensions 2–7
Enclosures 2–7
Panel Layout & Clearances 2–8
Agency Approvals 2–9
Environmental Specifications 2–9
Using Mounting Rails 2–10
Wiring Guidelines 2–11
Power Input Wiring 2–11
Fuse Protection for Input Power 2–11
External Power Source 2–12
Planning the Wiring Routes 2–13
Fuse Protection for Input and Output Circuits 2–13
I/O Point Numbering 2–13
System Wiring Strategies 2–14
PLC Isolation Boundaries 2–14
Powering I/O Circuits with the Auxiliary Supply 2–15
Powering I/O Circuits Using Separate Supplies 2–16
Connecting Operator Interface Devices 2–17
Connecting Programming Devices 2–17
Sinking / Sourcing Concepts 2–18
I/O “Common” Terminal Concepts 2–19
Connecting DC I/O to “Solid State” Field Devices 2–20
Solid State Input Sensors 2–20
Solid State Output Loads 2–20
Relay Output Wiring Methods 2–22
Surge Suppresion For Inductive Loads 2–23
Prolonging Relay Contact Life 2–24
DC Input Wiring Methods 2–25
DC Output Wiring Methods 2–26
High-Speed I/O Wiring Methods 2–27
F1–04SIM Input Simulator Wiring 2–28
Wiring Diagrams and Specifications 2–29
F1–130AR I/O Wiring Diagram 2–29
F1–130AR General Specifications 2–30
F1–130DR/F1–130DR–CE I/O Wiring Diagram 2–31
F1–130DR/F1–130DR–CE General Specifications 2–32
F1–130AD I/O Wiring Diagram 2–33
F1–130AD General Specifications 2–34
F1–130DD/F1–130DD–CE I/O Wiring Diagram 2–35
F1–130DD/F130–DD–CE General Specifications 2–36
F1–130AA I/O Wiring Diagram 2–37
F1–130AA General Specifications 2–38
F1–130DA I/O Wiring Diagram 2–39
F1–130DA General Specifications 2–40

F1-130DR-D I/O Wiring Diagram 2-41
F1-130DR-D General Specifications 2-42
F1-130DD-D I/O Wiring Diagram 2-43
F1-130DD-D General Specifications 2-44
Glossary of Specification Terms 2-45

Chapter 3: High-Speed Input and Pulse Output Features

Introduction 3-2
Built-in Motion Control Solution 3-2
Availability of HSIO Features 3-2
Dedicated High-Speed I/O Circuit 3-3
Wiring Diagrams for Each HSIO Mode 3-3
Choosing the HSIO Operating Mode 3-4
Understanding the Six Modes 3-4
Default Mode 3-4
Configuring the HSIO Mode 3-5
Configuring Inputs X0 – X3 3-5
Mode 10: High-Speed Counter 3-6
Purpose 3-6
Functional Block Diagram 3-6
Wiring Diagram 3-7
Interfacing to Counter Outputs 3-7
Setup for Mode 10 3-8
Presets and Special Relays 3-8
Preset Data Starting Location 3-9
Using Fewer than 24 Presets 3-9
Equal Relay Numbers 3-9
Calculating Your Preset Values 3-10
X Input Configuration 3-10
Writing Your Control Program 3-11
Program Example: Counter Without Preset 3-11
Program Example Cont'd 3-12
Counter With Presets Program Example 3-13
Counter With Preload Program Example 3-15
Troubleshooting Guide for Mode 10 3-16
Mode 20: Quadrature Counter 3-17
Purpose 3-17
Functional Block Diagram 3-17
Quadrature Encoder Signals 3-17
Wiring Diagram 3-18
Interfacing to Encoder Outputs 3-18
Setup for Mode 20 3-19
X Input Configuration 3-19
Writing Your Control Program 3-20
Quadrature Counter w/Preload Program Example 3-20
Program Example Cont'd 3-21
Counter Preload Program Example 3-22
Troubleshooting Guide for Mode 20 3-22
Mode 30: Pulse Output 3-23

Purpose 3–23
Functional Block Diagram 3–24
Wiring Diagram 3–25
Interfacing to Drive Inputs 3–25
Motion Profile Specifications 3–26
Physical I/O Configuration 3–26
Logical I/O Functions 3–26
Setup for Mode 30 3–27
Profile / Velocity Select Register 3–27
Profile Parameter Table 3–28
Trapezoidal Profile 3–28
Registration Profile 3–28
Velocity Profile 3–28
Choosing the Profile Type 3–29
Trapezoidal Profile Defined 3–29
Registration and Home Search Profiles Defined 3–29
Velocity Profile Defined 3–29
Trapezoidal Profile Operation 3–30
Trapezoidal Profile Applications 3–30
Trapezoidal Profile Program Example 3–31
Program Example Cont'd 3–32
Preload Position Value 3–32
Registration Profile Operation 3–33
Registration Applications 3–33
Registration Profile Program Example 3–34
Program Example Cont'd 3–35
Home Search Program Example 3–36
Velocity Profile Operation 3–38
Velocity Profile Applications 3–38
Velocity Profile Program Example 3–39
Program Example Cont'd 3–40
Pulse Output Error Codes 3–41
Troubleshooting Guide for Mode 30 3–41

Mode 40: High-Speed Interrupts 3–43

Purpose 3–43
Functional Block Diagram 3–43
Setup for Mode 40 3–44
Interrupts and the Ladder Program 3–44
External Interrupt Timing Parameters 3–45
Timed Interrupt Parameters 3–45
X Input / Timed INT Configuration 3–45
External Interrupt Program Example 3–46
Timed Interrupt Program Example 3–47

Mode 50: Pulse Catch Input 3–48

Purpose 3–48
Functional Block Diagram 3–48
Pulse Catch Timing Parameters 3–48
Setup for Mode 50 3–49
X Input Configuration 3–49
Pulse Catch Program Example 3–50

Mode 60: Discrete Inputs with Filter 3–51

Purpose 3–51
Functional Block Diagram 3–51
Input Filter Timing Parameters 3–51
Setup for Mode 60 3–52
X Input Configuration 3–52
Filtered Inputs Program Example 3–53

Chapter 4: CPU Specifications and Operation

Introduction 4–2
DL105 CPU Features 4–2
CPU Specifications 4–3
CPU Hardware Setup 4–4
Communication Port Pinout Diagrams 4–4
Connecting the Programming Devices 4–5
CPU Setup Information 4–5
CPU Modes 4–6
Mode of Operation at Power-up 4–6
Changing Modes in the DL105 PLC 4–6
Setting Bits in V7633 4–7
Auxiliary Functions 4–8
Clearing an Existing Program 4–9
Initializing System Memory 4–9
Setting Retentive Memory Ranges 4–9
Using a Password 4–10
CPU Operation 4–11
CPU Operating System 4–11
Program Mode 4–12
Run Mode 4–12
Read Inputs 4–13
Service Peripherals and Force I/O 4–13
Update Special Relays and Special Registers 4–13
Solve Application Program 4–14
Write Outputs 4–14
Diagnostics 4–14
I/O Response Time 4–15
Is Timing Important for Your Application? 4–15
Normal Minimum I/O Response 4–15
Normal Maximum I/O Response 4–16
Improving Response Time 4–17
CPU Scan Time Considerations 4–18
Reading Inputs 4–18
Writing Outputs 4–18
Application Program Execution 4–19
PLC Numbering Systems 4–20
PLC Resources 4–20
V–Memory 4–21
Binary-Coded Decimal Numbers 4–21
Hexadecimal Numbers 4–21

Memory Map **4-22**
Octal Numbering System 4-22
Discrete and Word Locations 4-22
V-Memory Locations for Discrete Memory Areas 4-22
Input Points (X Data Type) 4-23
Output Points (Y Data Type) 4-23
Control Relays (C Data Type) 4-23
Timers and Timer Status Bits (T Data type) 4-23
Timer Current Values (V Data Type) 4-24
Counters and Counter Status Bits (CT Data type) 4-24
Counter Current Values (V Data Type) 4-24
Word Memory (V Data Type) 4-25
Stages (S Data type) 4-25
Special Relays (SP Data Type) 4-25
DL105 System V-memory **4-26**
System Parameters and Default Data Locations (V Data Type) 4-26
DL105 Aliases **4-28**
DL105 Memory Map 4-29
X Input Bit Map **4-30**
Y Output Bit Map **4-30**
Control Relay Bit Map **4-30**
Stage Control / Status Bit Map **4-31**
Timer Status Bit Map **4-31**
Counter Status Bit Map **4-31**

Chapter 5: Standard RLL Instructions

Introduction **5-2**
Using Boolean Instructions **5-3**
END Statement 5-3
Simple Rungs 5-3
Normally Closed Contact 5-4
Contacts in Series 5-4
Midline Outputs 5-4
Parallel Elements 5-5
Joining Series Branches in Parallel 5-5
Joining Parallel Branches in Series 5-5
Combination Networks 5-6
Comparative Boolean 5-6
Boolean Stack 5-7
Immediate Boolean 5-8
Boolean Instructions **5-9**
Store (STR) 5-9
Store Not (STRN) 5-9
Or (OR) 5-10
Or Not (ORN) 5-10
And (AND) 5-11

And Not (ANDN) 5–11
And Store (AND STR) 5–12
Or Store (OR STR) 5–12
Out (OUT) 5–14
Or Out (OR OUT) 5–14
Positive Differential (PD) 5–15
Set (SET) 5–15
Reset (RST) 5–15
Set, Reset Instr. Continued 5–16
Pause (PAUSE) 5–16

Comparative Boolean 5–17

Store If Equal (STRE) 5–17
Store If Not Equal (STRNE) 5–17
Or If Equal (ORE) 5–18
Or If Not Equal (ORNE) 5–18
And If Equal (ANDE) 5–19
And If Not Equal (ANDNE) 5–19
Store (STR) 5–20
Store Not (STRN) 5–20
Or (OR) 5–21
Or Not (ORN) 5–21
And (AND) 5–22
And Not (ANDN) 5–22

Immediate Instructions 5–23

Store Immediate (STRI) 5–23
Store Not Immediate (STRNI) 5–23
Or Immediate (ORI) 5–23
Or Not Immediate (ORNI) 5–23
And Immediate (ANDI) 5–24
And Not Immediate (ANDNI) 5–24
Or Out Immediate (OROUTI) 5–25
Set Immediate (SETI) 5–26
Reset Immediate (RSTI) 5–26

Timer, Counter and Shift Register Instructions 5–27

Using Timers 5–27
Timer (TMR) and Timer Fast (TMRF) 5–28
Timer Example Using Discrete Status Bits 5–29
Timer Example Using Comparative Contacts 5–29
Accumulating Timer (TMRA) Accumulating Fast Timer (TMRAF) 5–30
Accumulating Timer Example using Discrete Status Bits 5–31
Accumulator Timer Example Using Comparative Contacts 5–31
Using Counters 5–32
Counter (CNT) 5–33
Counter Example Using Discrete Status Bits 5–34
Counter Example Using Comparative Contacts 5–34
Stage Counter (SGCNT) 5–35
Stage Counter Example Using Discrete Status Bits 5–36
Stage Counter Example Using Comparative Contacts 5–36
Up Down Counter (UDC) 5–37
Up / Down Counter Example Using Discrete Status Bits 5–38

Up / Down Counter Example Using Comparative Contacts 5–38
Shift Register (SR) 5–39
Accumulator / Stack Load and Output Data Instructions 5–40
Using the Accumulator 5–40
Copying Data to the Accumulator 5–40
Changing the Accumulator Data 5–40
Using the Accumulator Stack 5–42
Using Pointers 5–43
Load (LD) 5–45
Load Double (LDD) 5–46
Load Formatted (LDF) 5–47
Load Address (LDA) 5–48
Out (OUT) 5–49
Out Double (OUTD) 5–49
Out Formatted (OUTF) 5–50
Pop (POP) 5–50
Pop Instruction Continued 5–51
Logical Instructions (Accumulator) 5–52
And (AND) 5–52
And Double (ANDD) 5–53
Or (OR) 5–54
Or Double (ORD) 5–55
Exclusive Or (XOR) 5–56
Exclusive Or Double (XORD) 5–57
Compare (CMP) 5–58
Compare Double (CMPD) 5–59
Math Instructions 5–60
Add (ADD) 5–60
Add Double (ADDD) 5–61
Subtract (SUB) 5–62
Subtract Double (SUBD) 5–63
Multiply (MUL) 5–64
Divide (DIV) 5–65
Increment Binary (INCB) 5–66
Decrement Binary (DECB) 5–66
Bit Operation Instructions 5–67
Shift Left (SHFL) 5–67
Shift Right (SHFR) 5–68
Encode (ENCO) 5–69
Decode (DECO) 5–70
Number Conversion Instructions (Accumulator) 5–71
Binary (BIN) 5–71
Binary Coded Decimal (BCD) 5–72
Invert (INV) 5–73
Table Instructions 5–74
Move (MOV) 5–74
Move Memory Cartridge / Load Label (MOVMC), (LDLBL) 5–75
Copy Data From a Data Label Area to V–Memory 5–76
CPU Control Instructions 5–77

No Operation (NOP) 5–77
End (END) 5–77
Stop (STOP) 5–77
Program Control Instructions 5–78
Master Line Set (MLS) 5–78
Master Line Reset (MLR) 5–78
Understanding Master Control Relays 5–78
MLS/MLR Example 5–79
Interrupt Instructions 5–80
Interrupt (INT) 5–80
Interrupt Return (IRT) 5–80
Enable Interrupts (ENI) 5–80
Disable Interrupts (DISI) 5–80
External Interrupt Program Example 5–81
Timed Interrupt Program Example 5–82
Message Instructions 5–83
Fault (FAULT) 5–83
Fault Example 5–83
Data Label (DLBL) 5–84
ASCII Constant (ACON) 5–84
Numerical Constant (NCON) 5–84
Data Label Example 5–85

Chapter 6: Drum Instruction Programming

Introduction 6–2
Purpose 6–2
Drum Terminology 6–2
Drum Chart Representation 6–3
Output Sequences 6–3
Step Transitions 6–4
Drum Instruction Parameters 6–4
Timer-Only Transitions 6–4
Timer and Event Transitions 6–5
Event-Only Transitions 6–6
Counter Assignments 6–6
Last Step Completion 6–7
Overview of Drum Operation 6–8
Drum Instruction Block Diagram 6–8
Powerup State of Drum Registers 6–9
Drum Control Techniques 6–10
Drum Control Inputs 6–10
Self-Resetting Drum 6–11
Initializing Drum Outputs 6–11
Using Complex Event Step Transitions 6–11
Drum Instruction 6–12
Event Drum (EDRUM) 6–12
Handheld Programer Drum Mnemonics 6–14

Chapter 7: RLL*PLUS* Stage Programming

Introduction to Stage Programming **7-2**
Overcoming “Stage Fright” 7-2
Learning to Draw State Transition Diagrams **7-3**
Introduction to Process States 7-3
The Need for State Diagrams 7-3
A 2-State Process 7-3
RLL Equivalent 7-4
Stage Equivalent 7-4
Let’s Compare 7-5
Initial Stages 7-5
What Stage Bits Do 7-6
Stage Instruction Characteristics 7-6
Using the Stage Jump Instruction for State Transitions **7-7**
Stage Jump, Set, and Reset Instructions 7-7
Stage Program Example: Toggle On/Off Lamp Controller **7-8**
A 4-State Process 7-8
Four Steps to Writing a Stage Program **7-9**
Stage Program Example: A Garage Door Opener **7-10**
Garage Door Opener Example 7-10
Draw the Block Diagram 7-10
Draw the State Diagram 7-11
Add Safety Light Feature 7-12
Modify the Block Diagram and State Diagram 7-12
Using a Timer Inside a Stage 7-13
Add Emergency Stop Feature 7-14
Exclusive Transitions 7-14
Stage Program Design Considerations **7-15**
Stage Program Organization 7-15
How Instructions Work Inside Stages 7-16
Using a Stage as a Supervisory Process 7-17
Stage Counter 7-17
Power Flow Transition Technique 7-18
Stage View in DirectSOFT 7-18
RLL*PLUS* Stage Instructions **7-19**
Stage (SG) 7-19
Initial Stage (ISG) 7-20
JUMP (JMP) 7-20
Questions and Answers about Stage Programming **7-21**

Chapter 8: Maintenance and Troubleshooting

Hardware System Maintenance **8-2**
Diagnostics **8-2**
CPU Indicators **8-6**

Communications Problems **8–7**
I/O Point Troubleshooting **8–8**
Noise Troubleshooting **8–10**
Machine Startup and Program Troubleshooting **8–11**

Appendix A: Auxiliary Functions

Introduction **A–2**
Purpose of Auxiliary Functions A–2
Accessing AUX Functions via DirectSOFT A–3
Accessing AUX Functions via the Handheld Programmer A–3
AUX 2* — RLL Operations **A–4**
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
AUX 3* — V-memory Operations **A–4**
AUX 31 Clear V–Memory A–4
AUX 4* — I/O Configuration **A–4**
AUX 41 Show I/O Configuration A–4
AUX 5* — CPU Configuration **A–5**
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 57 Set Retentive Ranges A–5
AUX 58 Test Operations A–6
AUX 5B Counter Interface Configuration A–6
AUX 6* — Handheld Programmer Configuration **A–6**
AUX 61 Show Revision Numbers A–6
AUX 62 Beeper On / Off A–6
AUX 65 Run Self Diagnostics A–6
AUX 7* — EEPROM Operations **A–7**
Transferrable Memory Areas A–7
AUX 71 CPU to HPP EEPROM A–7
AUX 72 HPP EEPROM to CPU A–7
AUX 73 Compare HPP EEPROM to CPU A–7
AUX 74 HPP EEPROM Blank Check A–7
AUX 75 Erase HPP EEPROM A–7
AUX 76 Show EEPROM Type A–7
AUX 8* — Password Operations **A–8**
AUX 81 Modify Password A–8
AUX 82 Unlock CPU A–8
AUX 83 Lock CPU A–8

Appendix B: DL105 Error Codes

DL105 Error Codes B–2

Appendix C: Instruction Execution Times

Introduction C–2
V-Memory Data Registers C–2
V-Memory Bit Registers C–2
How to Read the Tables C–2
Instruction Execution Times C–3
Boolean Instructions C–3
Comparative Boolean Instructions C–3
Immediate Instructions C–6
Timer, Counter, Shift Register, EDRUM Instructions C–6
Accumulator Data Instructions C–7
Logical Instructions C–8
Math Instructions C–8
Bit Instructions C–9
Number Conversion Instructions C–9
Table Instructions C–9
CPU Control Instructions C–10
Program Control Instructions C–10
Interrupt Instructions C–10
Message Instructions C–10
RLL*PLUS* Instructions C–10

Appendix D: Special Relays

DL105 PLC Special Relays D–2
Startup and Real-Time Relays D–2
CPU Status Relays D–2
System Monitoring D–2
Accumulator Status D–3
HSIO Pulse Catch Relay D–3
Equal Relays for HSIO Mode 10 Counter Presets D–3

Appendix E: PLC Memory

DL105 PLC Memory E–2

Appendix F: European Union Directives (CE)

European Union (EU) Directives F–2
Member Countries F–2
Special Installation Manual F–4
Other Sources of Information F–4
Basic EMC Installation Guidelines F–5
Enclosures F–5
AC Mains Filters F–5
Suppression and Fusing F–5

Internal Enclosure Grounding F–6
Equi–potential Grounding F–6
Communications and Shielded Cables F–6
Analog and RS232 Cables F–7
Multidrop Cables F–7
Shielded Cables F–7
within Enclosures F–7
Network Isolation F–7

Appendix G: Numbering Systems

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

Index

1

Getting Started

In This Chapter. . . .

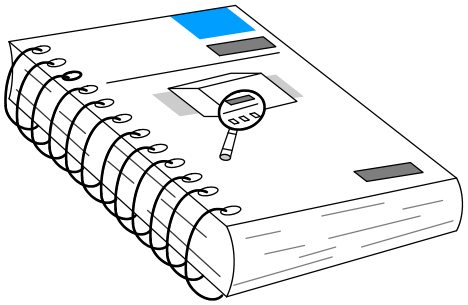
- Introduction
- Conventions Used
- DL105 Micro PLC Components
- Programming Methods
- I/O Selection Quick Chart
- Quick Start for PLC Checkout and Programming
- Steps to Designing a Successful System
- Questions and Answers about DL105 Micro PLCs

Introduction

The Purpose of this Manual

Thank you for purchasing a DL105 Micro PLC. This manual shows you how to install, program, and maintain all the Micro PLCs in the DL105 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 DL105 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 DL105 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 DL105 customer, we suggest you read this manual completely so you can understand the wide variety of features in the DL105 family of products. We believe you will be pleasantly surprised with how much you can accomplish with **AutomationDirect** products.

Supplemental Manuals

If you have purchased operator interfaces or ***Direct***SOFT 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 Web site is: **http://www.automationdirect.com**

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

Conventions Used



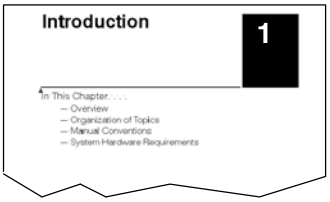
When you see the “notepad” icon in the left-hand margin, the paragraph to its immediate right will be a **special note**.
The word **NOTE:** in boldface will mark the beginning of the text.



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

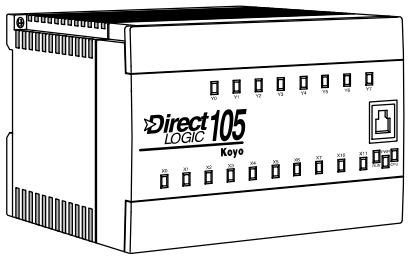
Key Topics for Each Chapter

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



DL105 Micro PLC Components

The DL105 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 in larger, more expensive systems. These include removeable connectors, RS-232C communication port, and +24V auxiliary power supply.



The DL105 Micro PLC Family

The DL105 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 DL230 CPU, plus several more instructions specifically designed for machine control applications. All DL105 PLCs have an RS–232C communications port, and the AC-powered versions have an auxiliary +24V output. Units with DC inputs have selectable high-speed input features on four input points. Units with DC outputs offer selectable pulse output capability on the first two output points. All DL105 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 4, CPU Specifications and Operation. The eight types of DL105 Micro PLCs provide a variety of Input/Output choices, listed in the following table.

DL105 Part Number	Discrete Input Type	Discrete Output Type	External Power	Auxiliary 24V Output	High-Speed Input	Pulse Output
F1–130AR	AC	Relay	94–240 VAC	Yes	No	No
F1–130DR	DC	Relay	94–240 VAC	Yes	Yes	No
F1–130AD	AC	DC	94–240 VAC	Yes	No	Yes
F1–130DD	DC	DC	94–240 VAC	Yes	Yes	Yes
F1–130AA	AC	AC	94–240 VAC	Yes	No	No
F1–130DA	DC	AC	94–240 VAC	Yes	Yes	No
F1–130DR–D	DC	Relay	10–30 VDC	No	Yes	No
F1–130DD–D	DC	DC	10–30 VDC	No	Yes	Yes

Programming Methods

Two programming methods are available: a software programming package and a handheld programmer. The software and the handheld programmer support RLL (Relay Ladder Logic) and RLL*PLUS* (STAGE) programming.

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

The DL105 Micro PLC can be programmed with one of the most advanced programming packages in the industry --***Direct*SOFT5. This is the most recent version of ***Direct*SOFT, a Windows-based software package that supports familiar features such as cut-and-paste between applications, point-and-click editing, viewing and editing multiple application programs at the same time, etc. ***Direct*SOFT universally supports the ***Direct*LOGIC CPU families. This means that you can use the same ***Direct*SOFT package to program DL05, DL06, DL105, DL205, DL305, DL405 or any new CPU we may add to our product line. (Upgrade software may be required for new CPUs as they become available.)

There is a separate manual that discusses the ***Direct***SOFT programming software which is included with your software package.

Handheld Programmer

All DL105 Micro PLCs have a built-in programming port for use with the handheld programmer (D2-HPP), the same programmer used with the DL205 family. The handheld programmer can be used to create, modify and debug your application program. A separate manual discusses the Handheld Programmer.

I/O Quick Chart Selection

The eight versions of the DL105 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 DL105 Micro PLC to interface to the field devices in your application.

DL105 Part Number	INPUTS			OUTPUTS		
	I/O type / commons	Sink / Source	Voltage Ranges	I/O type / commons	Sink / Source	Voltage / Current Ratings
F1-130AR	AC / 3	-	80 - 132 VAC 90 - 150 VDC	Relay / 4	Sink or Source	12 - 30 VDC, 7A * 12 - 250 VAC, 7A *
F1-130DR	DC / 3	Sink or Source	10 - 26.4 VDC 21.6 - 26.4 VAC	Relay / 4	Sink or Source	12 - 30 VDC, 7A * 12 - 250 VAC, 7A *
F1-130AD	AC / 3	-	80 - 132 VAC 90 - 150 VDC	DC / 1 **	Sink	5 - 30 VDC, 0.3A (Y0-Y1) 5 - 30 VDC, 0.6A (Y3-Y7)
F1-130DD	DC / 3	Sink or Source	10 - 26.4 VDC 21.6 - 26.4 VAC	DC / 1 **	Sink	5 - 30 VDC, 0.3A (Y0-Y1) 5 - 30 VDC, 0.6A (Y3-Y7)
F1-130AA	AC / 3	-	80 - 132 VAC 90 - 150 VDC	AC / 4	-	20 - 140 VAC, 47 - 63 Hz 1.7A *
F1-130DA	DC / 3	Sink or Source	10 - 26.4 VDC 21.6 - 26.4 VAC	AC / 4	-	20 - 140 VAC, 47 - 63 Hz 1.7A *
F1-130DR-D	DC / 3	Sink or Source	10 - 26.4 VDC 21.6 - 26.4 VAC	Relay / 4	Sink or Source	12 - 30 VDC, 7A * 12 - 250 VAC, 7A *
F1-130DD-D	DC / 3	Sink or Source	10 - 26.4 VDC 21.6 - 26.4 VAC	DC / 1 **	Sink	5 - 30 VDC, 0.3A (Y0-Y1) 5 - 30 VDC, 0.6A (Y3-Y7)

* Subject to temperature derating chart. See Chapter 2 Specifications for your particular DL105 version.

** DC outputs have one electrical common, but it is accessible at three terminals on the output connector.

Quick Start for PLC Checkout and Programming

If you have experience with PLCs, or if you just want to setup a quick example, this example is for you! 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.

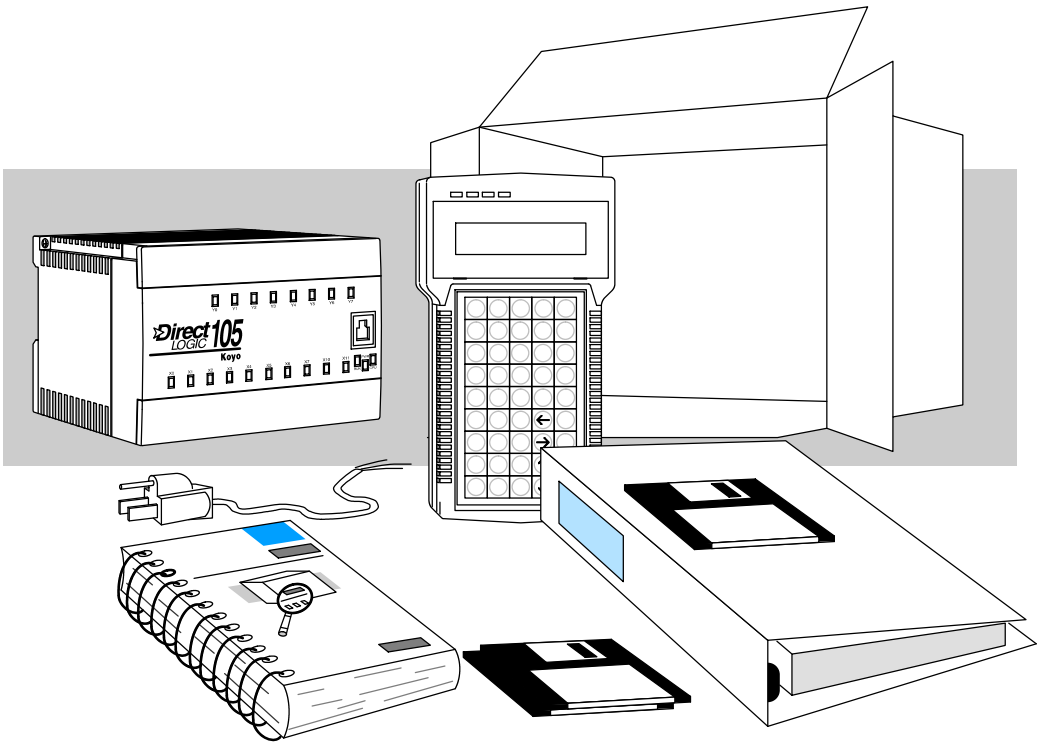
Step 1: Unpack the Recommended DL105 Equipment

Unpack the DL105 equipment and verify you have the recommended parts necessary to build this demonstration system. The components are:

- DL105 Micro PLC
- AC power cord for AC-powered units
- F1-04SIM input simulator, or toggle switches (see Step 2 on next page).
- Hook-up wire, 16-20 AWG
- DL105 User Manual (this manual)
- A small screwdriver, regular or #2 Philips type

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

- ***Direct***SOFT Programming Software, ***Direct***SOFT Manual, and a programming cable (connects the DL105 to a personal computer),or
- D2-HPP Handheld Programmer (comes with programming cable), and the Handheld Programmer Manual

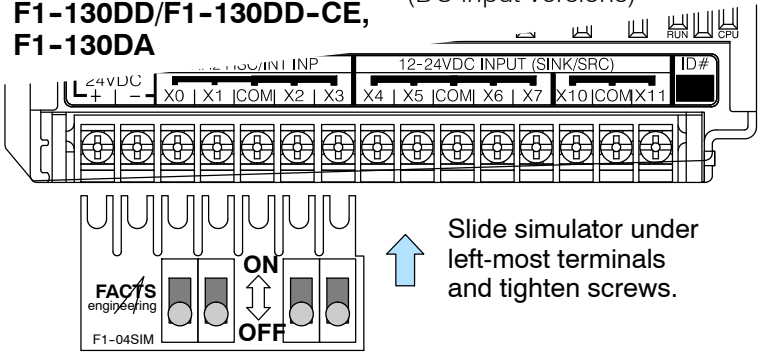


Step 2: Connect Switches to Input Terminals

To finish this quick-start exercise or study other examples in this manual, you'll need to connect some input switches as shown below. For most models, the F1-04SIM Input Simulator is a quick way to install four switches on inputs X0 – X3. DC-powered units will require routing DC power to the simulator as shown. We recommend using one of the models compatible with the input simulator as you learn the DL105.

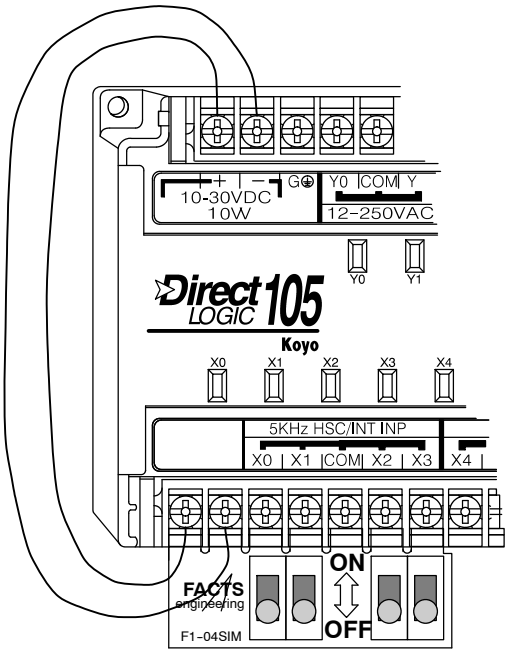
However, you may wire individual toggle switches to AC-powered units as shown, *as long as you follow the instructions in the accompanying WARNING note.*

F1-130DR/F1-130DR-CE, F1-130DD/F1-130DD-CE, F1-130DA (DC input versions)

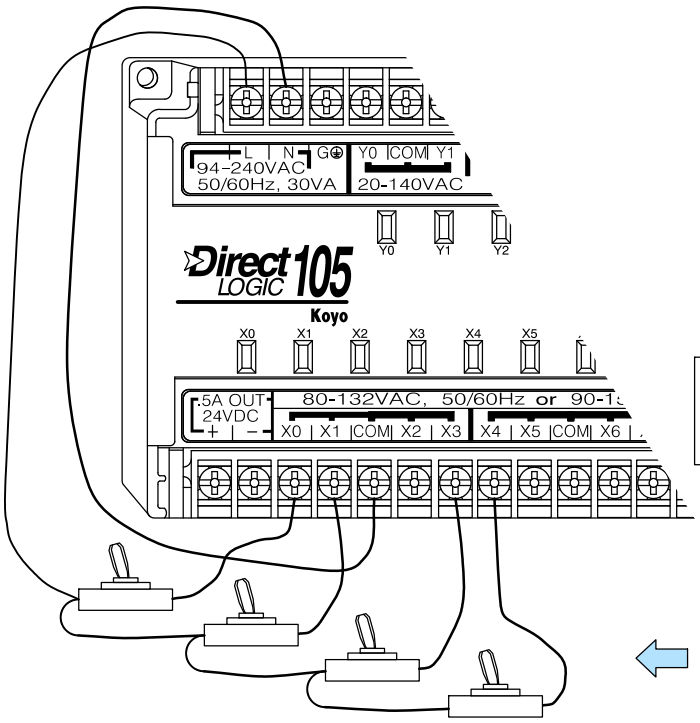


Slide simulator under left-most terminals and tighten screws.

F1-130DR-D, F1-130DD-D
(DC-powered versions)



F1-130AR, F1-130AD, F1-130AA
(AC input versions, 120VAC only)



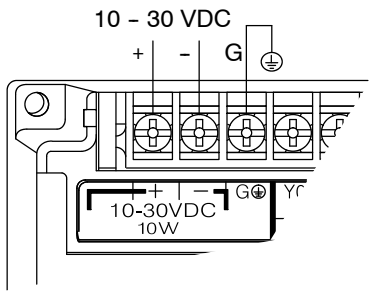
Toggle Switches, UL Listed

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 DL105 when wiring the switches. Only use UL-approved switches rated for at least 250VAC, 1A. Firmly mount the switches before using.
NEVER use the input simulator on these units with AC-type discrete inputs.

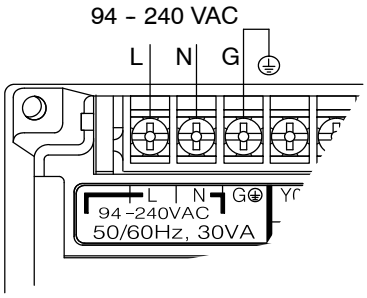
Step 3: Connect the Power Wiring

Connect the power input wiring for the version DL105 you have. 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.

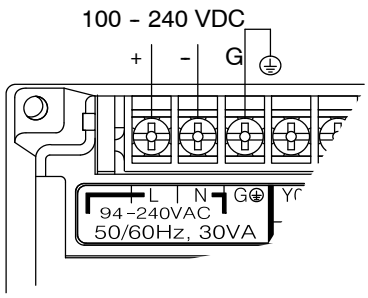
12/24 VDC Power Input



110/220 VAC Power Input

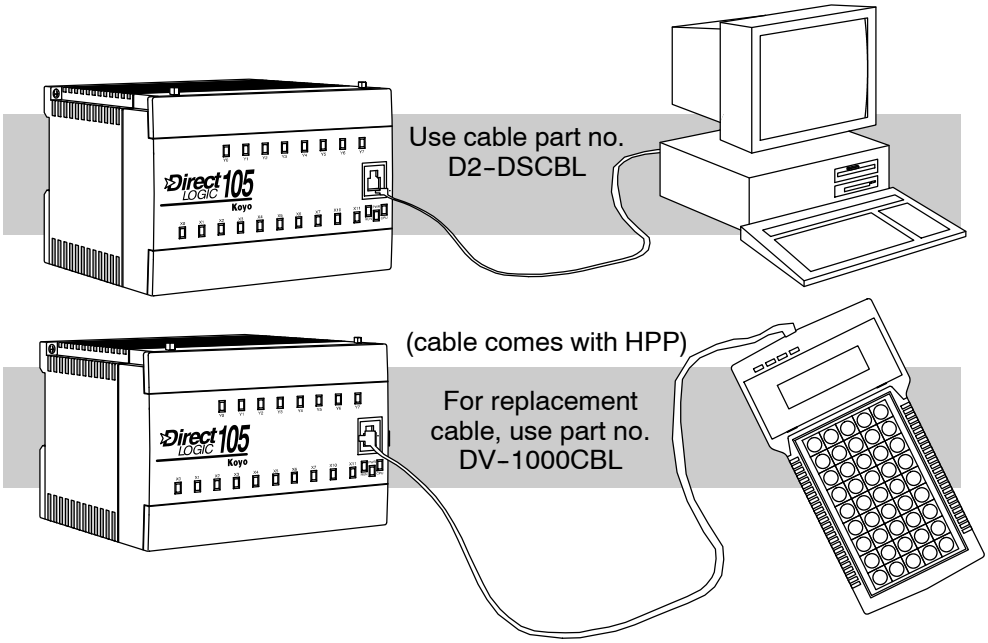


125 VDC Power Input



Step 4: Connect the Programming Device

Most programmers will use ***Direct***SOFT programming software, installed on a personal computer. Or, you may need the portability of the Handheld Programmer. Both devices will connect the COM1 port of the DL105 via the appropriate cable.



Step 5: Switch on the System Power

Apply power to the system and ensure the PWR indicator on the DL105 is on. If not, remove power from the system and check all wiring and refer to the troubleshooting section in Chapter 8 for assistance.

Step 6: Initialize Scratchpad Memory

It's a good precaution to always clear the system memory (scratchpad memory) on a new DL105. When a unit has been without power for several days, the system RAM contents may have been corrupted and will require initialization.

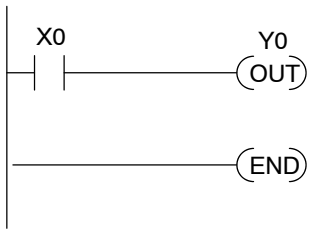
- In ***Direct***SOFT, select the PLC **Menu > Setup > Initialize Scratchpad**. For additional information, see the ***Direct***SOFT Manual.
- For the Handheld Programmer, use the AUX key and execute AUX 54. For additional information, see the Handheld Programmer Manual.

Step 7: Enter a Ladder Program

At this point, ***Direct***SOFT programmers need to refer to the Quick Start Tutorial in the ***Direct***SOFT Manual. There you will learn how to establish a communications link with the DL105 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 DL105 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.

Equivalent ***Direct***SOFT display



CLR	CLR					Clear the Program
C 2	E 4	AUX	ENT	ENT	CLR	
NEXT	$ STR	→	A 0	ENT		Move to the first address and enter X0 contact
GX OUT	→	A 0	ENT			Enter output Y0
SHFT	E 4	N TMR	D 3	ENT		Enter the END statement

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

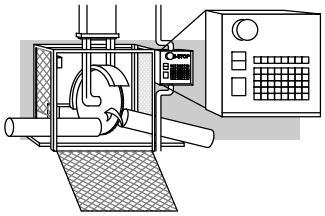
The RUN indicator on the PLC will illuminate indicating the CPU has entered the Run mode. If not, repeat this step, ensuring the program is entered properly or refer to the troubleshooting guide in chapter 8.

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

Steps to Designing a Successful System

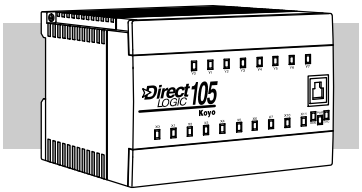
Step 1: Review the Installation Guidelines

Always make safety the first priority in any system design. Chapter 2 provides several guidelines that will help you design a safer, more reliable system. This chapter also includes wiring guidelines for the various versions of the DL105 PLC.



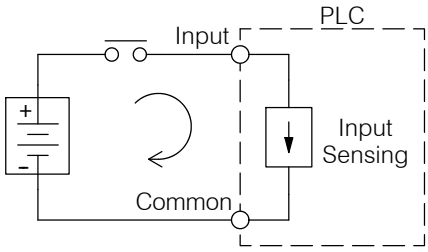
Step 2: Understand the PLC Setup Procedures

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



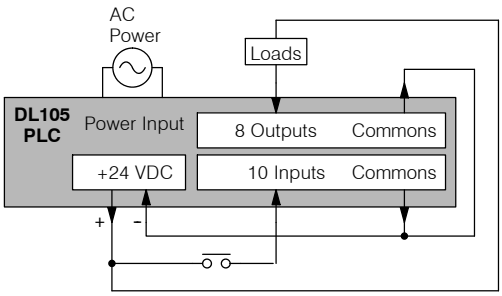
Step 3: Review the I/O Selection Criteria

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.



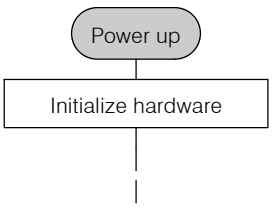
Step 4: Choose a System Wiring Strategy

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 DL105 system processes information. This involves not only program execution steps, but also involves the various modes of operation and memory layout characteristics.

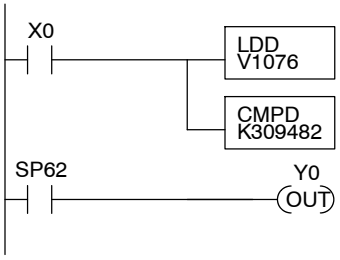


Step 6: Review the Programming Concepts

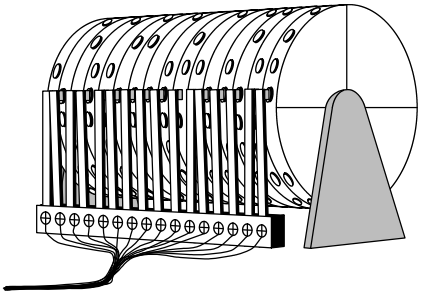
The DL105 PLC instruction set provides for three main approaches to solving the application program, depicted in the figure below.

- 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 EDRUM instruction is best for a repetitive process based on a single series of steps.
- Stage programming (also called RLL*PLUS*) is based on state-transition diagrams. Stages divide the ladder program into sections which correspond to the states in a flow chart you draw for your process.

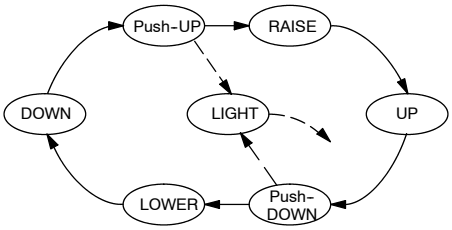
Standard RLL Programming
(see Chapter 5)



Timer/Event Drum Sequencer
(see Chapter 6)



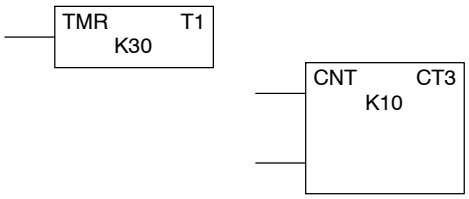
Stage Programming
(see Chapter 7)



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

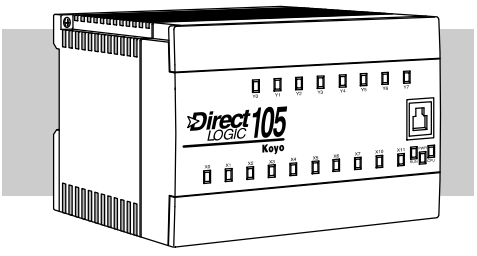
Step 7: Choose the Instructions

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 DL105 Micro PLC has many built-in features such as error codes that can help you quickly identify problems.



Questions and Answers about DL105 Micro PLCs

Q. What is the instruction set like?

A. The instruction set is very close to our popular DL230 CPU. However, there are significant additions, such as the drum instruction and High-Speed I/O capability.

Q. Do I have to buy the full *Direct*SOFT5 programming package to program the DL105?

A. No, ***Direct***SOFT100® can be used. It has the same features as the full version of ***Direct***SOFT5 but allows only 100 words of ladder code that can be downloaded to the PLC. It will also program DL05/06/205/305/405 systems, and it is only available for download online. (Order programming cable separately.) 32-bit application, Windows 2000 or Windows XP (Pro or Home) recommended. This software is available at no cost to you.

Q. Is the DL105 networkable or expandable?

A. No, the DL130 series is stand-alone PLCs. However, our DL205 system is expandable and networkable (with DL240 CPU), yet very compact and affordable.

Q. Does the DL105 have motion control capability?

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

Q. Are the ladder programs stored in a removable EPROM?

A. The DL105 contains a non-removable EEPROM for program storage, which may be written and erased thousands of times. You may transfer programs to/from ***Direct***SOFT on a PC, or the HPP (which does support a removable EEPROM).

Q. Does the DL105 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 DL105 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. Can the DL105 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.

Q. Which devices can I connect to the Com1 port of the DL105?

A. The port is RS-232C, fixed at 9600 baud, and uses the proprietary K-sequence protocol. The port communicates with the following devices:

- DV-1000 Data Access Unit or Optimation Operator interface panels
- ***Direct***SOFT (running on a personal computer)
- D2-HPP handheld programmer
- Other devices which communicate via K-sequence protcol should work with the DL105 Micro PLC. Contact the vendor for details.

2

Installation, Wiring, and Specifications

In This Chapter. . . .

— Safety Guidelines
— Orientation to DL105 Front Panel
— Mounting Guidelines
— Wiring Guidelines
— System Wiring Strategies
— Wiring Diagrams and Specifications
— Glossary of Specification Terms

Safety Guidelines



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.

Safety Techniques

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 may help reduce the risk of safety concerns.

- Orderly system shutdown sequence in the PLC control program.
- System power disconnects (guard limits, emergency stop switches, etc.)

Three Levels of Protection

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

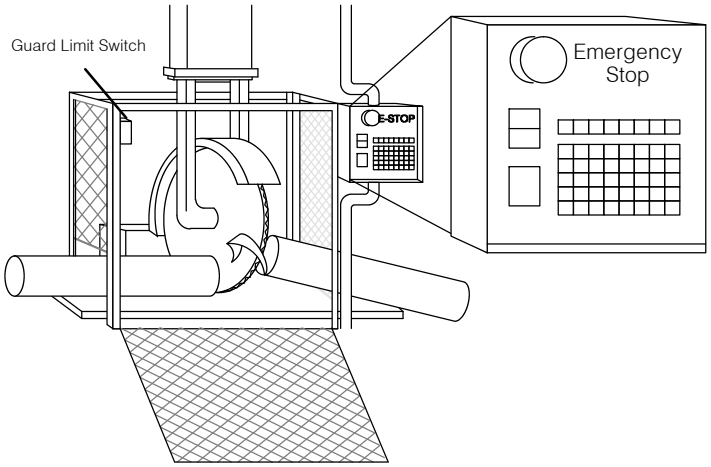
- Orderly system shutdown sequence in the PLC control program
- Mechanical disconnect for output module power
- Emergency stop switch for disconnecting system power

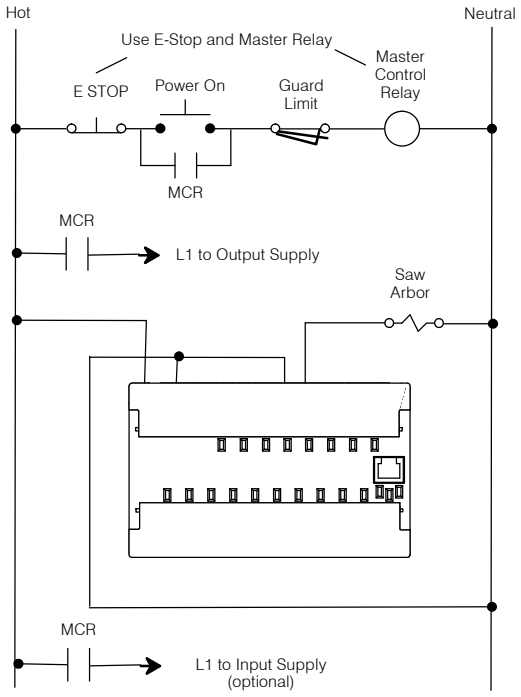
Emergency Stops

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.).





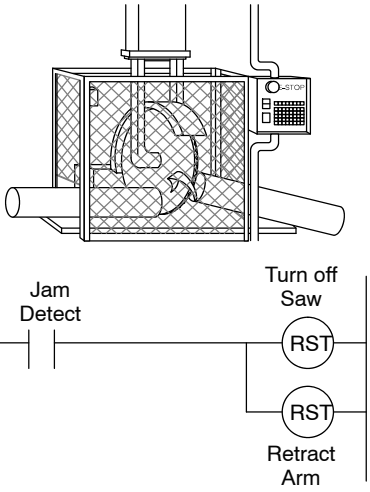
Emergency Power Disconnect

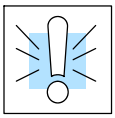
A properly rated emergency power disconnect should be used to power the PLC controlled system as ameans of removing the power from the entire control system. It may be necessary to install a capacitor across the disconnect to protect against a condition known as “outrush“. This condition occurs when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

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

Orderly System Shutdown

Ideally, the first level of protection can be provided with the PLC control program by identifying machine problems. Analyze your application and identify any shutdown sequences that must be performed. Typical problems such as jammed or missing parts, empty bins, etc., create a risk of personal injury or equipment damage.

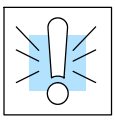




WARNING: The control program *must not* be the only form of protection for any problems that may result in a risk of personal injury or equipment damage.

Class 1, Division 2 Approval

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

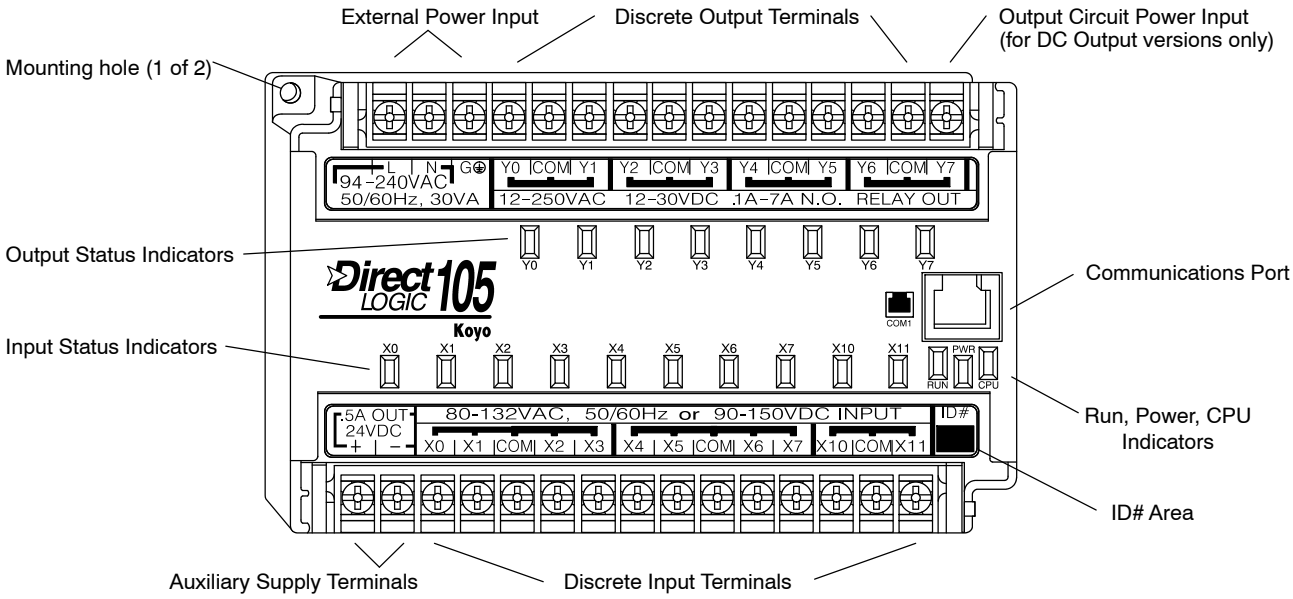


WARNING: Explosion Hazard! Substitution of components may impair suitability for Class 1, Division 2.

Do not disconnect equipment unless power has been switched off or area is known to be non-hazardous.

Orientation to DL105 Front Panel

All connections, indicators, and labels on the DL105 Micro PLCs are located on its front panel. Please refer to the drawing below.

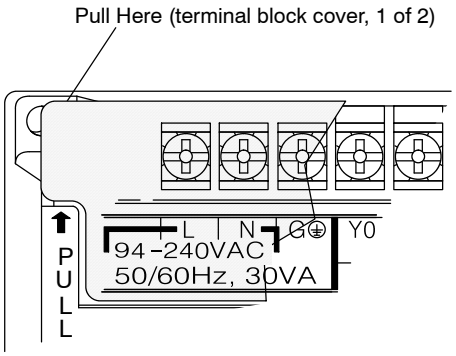


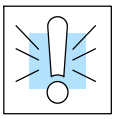
The upper connector accepts external power connections on the left-most terminals. The remainder of the terminals accept wires for the eight output points and their commons. On DC output versions, the end terminal on the right accepts power for the output stage. In many applications the external power to the Micro PLC also powers the loads and output circuit, so this terminal block groups them together.

The lower connector delivers the internal +24VDC auxiliary supply output on the two left-most terminals (AC-powered units only). On DC-powered units, the terminals are not used. The remaining terminals accept wires for the ten discrete inputs and their commons. In many applications the auxiliary +24VDC output also powers the input circuit, so this terminal block groups them together.

Accessing the I/O Terminals

To access the terminals just pull forward on the corner of the terminal cover where it is marked “pull”, as shown to the right. After exposing the connector block, it may be removed from the unit if desired.





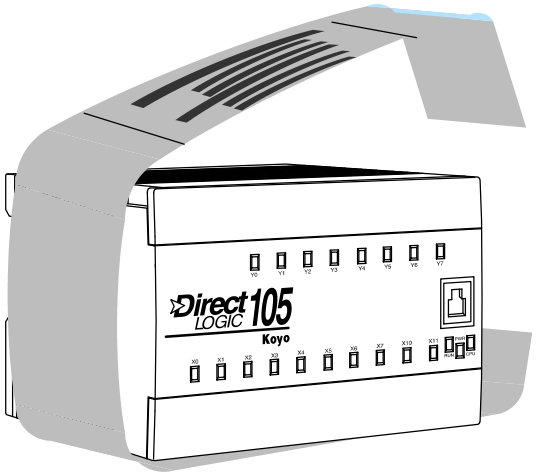
WARNING: For some applications, field device power may still be present on the terminal block even though the Micro PLC is turned off. To minimize the risk of electrical shock, check all field device power *before* you expose or remove either connector. Be sure to leave the covers normally closed.

Protective Sheet for DL105 Vents

Some machine fabrication environments may accidentally cause conductive debris to fall through the DL105 cooling vents and into the unit. All DL105 units come with a protective sheet wrapped around the unit, covering the cooling vents. However, it must be removed before electrical operation. Just unfasten the sheet on the right side of the unit. The instructions are reprinted below for your reference.

CAUTION

1. DO NOT REMOVE THIS PROTECTIVE SHEET until installation and wiring are completed.
2. REMOVE THIS SHEET before operation to enable heat to escape for proper cooling.



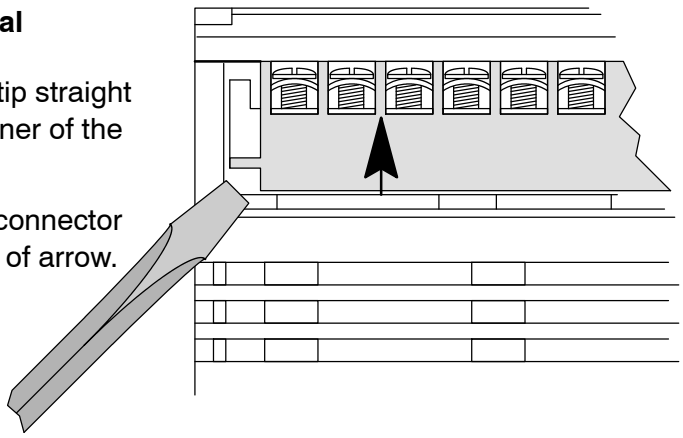
Connector Removal

The input and output terminal block connectors on the DL105 are identical. The connectors are designed for easy removal with just a small screwdriver. The drawing below shows the procedure for removal at one end. You'll need to work at both ends of the connector, so the ends move upwards approximately together.

Connector Removal

1. Insert the screwdriver tip straight into the opening at the corner of the connector as shown.

2. Pry screwdriver tip so connector moves upward in direction of arrow.



The two terminal block connectors on DL105 PLCs have regular screw terminals, which will accept either standard or #2 Philips screwdriver tips. You can insert one 14 AWG wire under a terminal, or two 16 AWG wires (one on each side of the screw). Be careful not to overtighten; maximum torque is 6 inch/ounces.

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

Part Number	Qty Per Package	Description
F1-IOCON	4	DL105 I/O Terminal Block
F1-IOCVR	4	DL105 I/O Terminal Cover

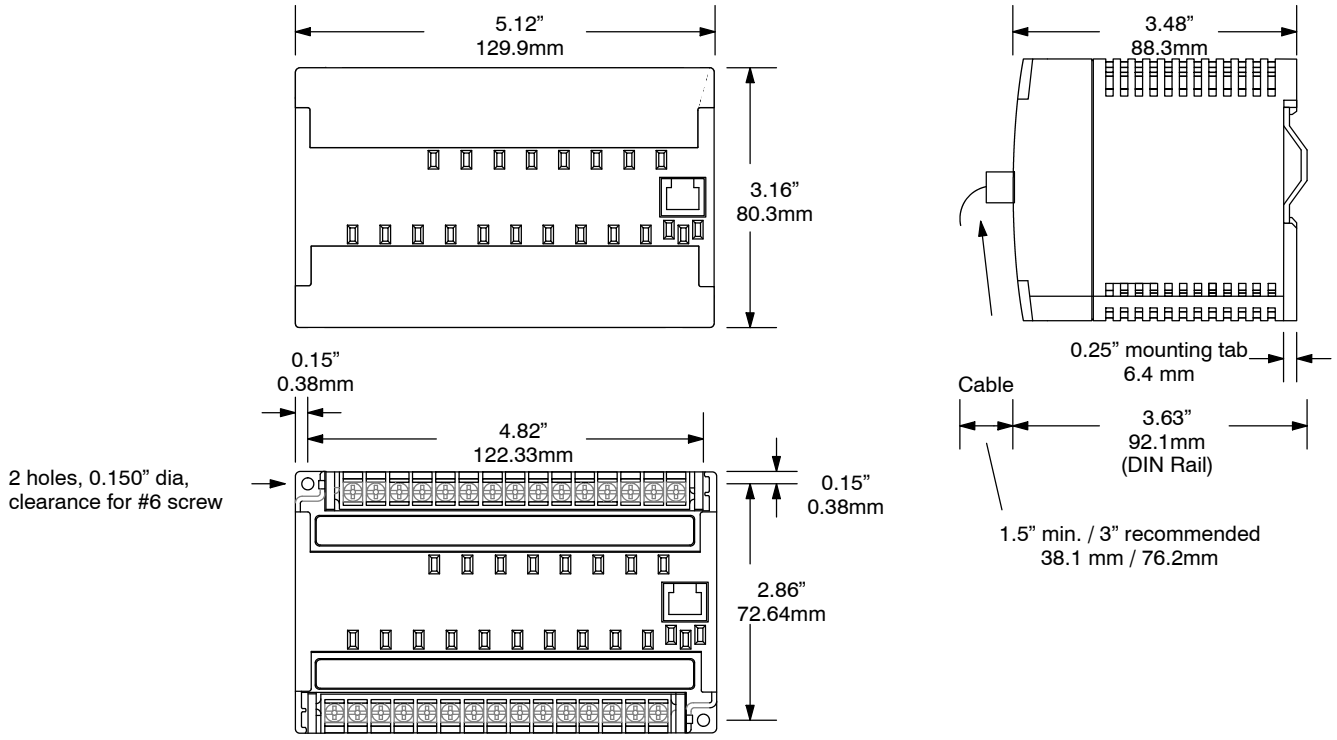
Mounting Guidelines

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 DL105. Make sure you follow the installation guidelines to allow proper spacing from other components.



Enclosures

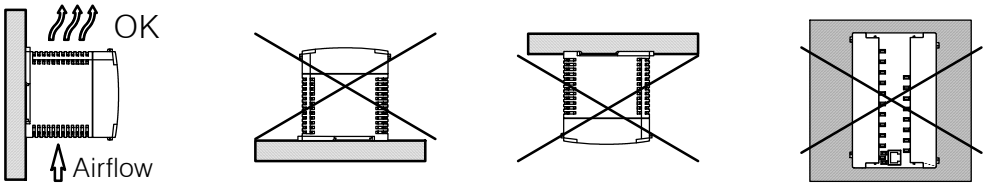
Your selection of a proper enclosure is important to ensure safe and proper operation of your DL105 system. Applications of DL105 systems vary and may require additional features. The minimum considerations for enclosures include:

- 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

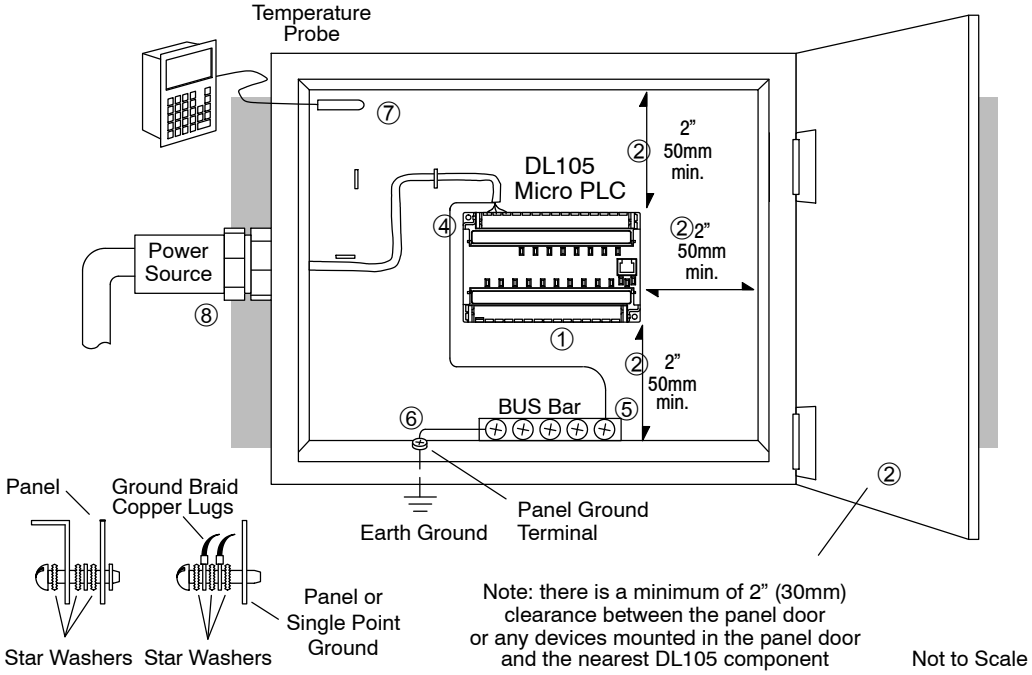
Panel Layout & Clearances

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 bases horizontally as shown below to provide proper ventilation. You ***cannot*** mount the DL105 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 DL105 base must be connected to a single point ground. Use copper stranded wire to achieve a low impedance. Copper eye lugs should be crimped and soldered to the ends of the stranded wire to ensure good surface contact.
5. There must be a single point ground (i.e. copper bus bar) for all devices in the panel requiring an earth ground return. The single point of ground must be connected to the panel ground termination. The panel ground termination must be connected to earth ground. Minimum wire sizes, color coding, and general safety practices should comply with appropriate electrical codes and standards for your area.

6. A good common ground reference (Earth ground) is essential for proper operation of the DL105. One side of all control and power circuits and the ground lead on flexible shielded cable must be properly connected to Earth ground. There are several methods of providing an adequate common ground reference, including:
   a) Installing a ground rod as close to the panel as possible.
   b) Connection to incoming power system ground.
7. Evaluate any installations where the ambient temperature may approach the lower or upper limits of the specifications. If you suspect the ambient temperature will not be within the operating specification for the DL105 system, measures such as installing a cooling/heating source must be taken to get the ambient temperature within the range of specifications.
8. The DL105 systems are designed to be powered by 110 VAC , 220 VAC, 125 VDC or 24 VDC normally available throughout an industrial environment. Isolation transformers and noise suppression devices are not normally necessary, but may be helpful in eliminating/reducing suspected power problems.



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.

Agency Approvals

Some applications require agency approvals for particular components. The DL105 Micro PLC approval are listed below:

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

Environmental Specifications

The following table lists the environmental specifications that generally apply to DL105 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 DL105.

Specification	Rating
Storage temperature	-4° F to 158° F (-20° C to 70° C)
Ambient operating temperature*	32° F to 140° F (0° C to 60° C)
Ambient humidity**	5% - 95% relative humidity (non-condensing)
Vibration resistance	MIL STD 810C, Method 514.2
Shock resistance	MIL STD 810C, Method 516.2
Noise immunity	NEMA (ICS3-304)
Atmosphere	No corrosive gases

* 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° to70° 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.

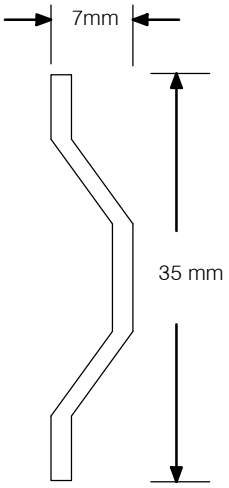
Using Mounting Rails

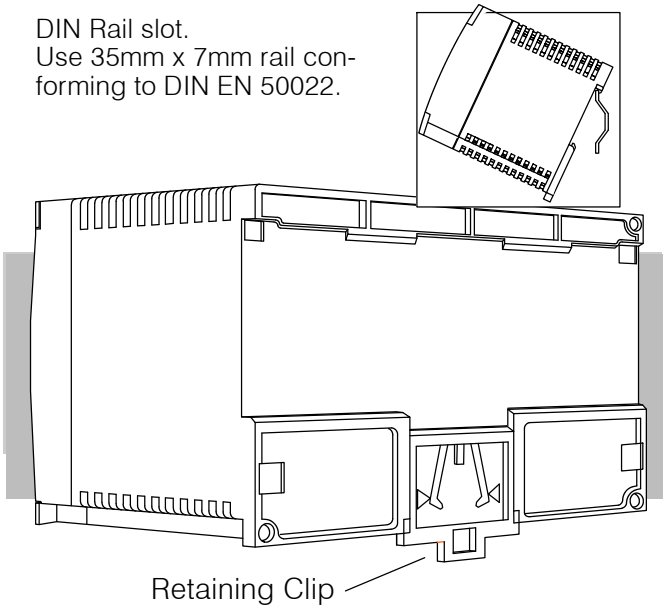
DL105 Micro PLCs can also be secured to a cabinet by using mounting rails. We recommend rails that conform to DIN EN standard 50 022. They are approximately 35mm high, with a depth of 7.5mm. If you mount the Micro PLC on a rail, do consider using end brackets on each side of the PLC. The end bracket helps keep the PLC from sliding horizontally along the rail, reducing the possibility of accidentally pulling the wiring loose. These items can be ordered from **AutomationDirect.com** website, or from our catalog.

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





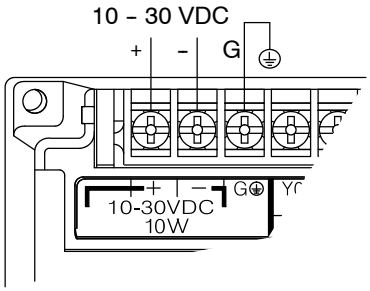
Wiring Guidelines

Power Input Wiring

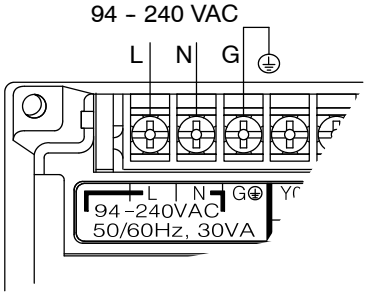
The diagram shows various possible external power connections for DL105 Micro PLCs. The terminals can accept up to 14 AWG wire. You may be able to use larger wiring depending on the wire type, but 14 AWG is the recommended size.

NOTE: You can connect either 115 VAC, 220 VAC, or 125 VDC to AC-powered versions of the DL105. Special wiring or jumpers are not required as with some of the other ***Direct***LOGIC products.

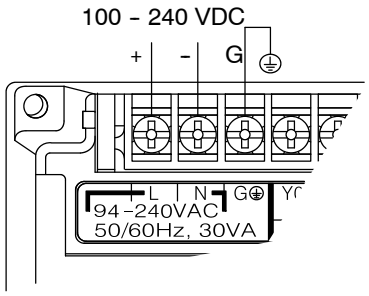
12/24 VDC Power Input



110/220 VAC Power Input



125 VDC Power Input



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/CSA 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 0.375A (for example, a Littlefuse 312.375 or equivalent).

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 0.5A (for example, a Littlefuse 312.500 or equivalent).

125 VDC Operation

Proper fusing techniques are required when operating from 125 VDC. Depending on your ground reference, the hot lead must be fused. A DC failure can maintain an arc for much longer time and distance. Typically, the main bus is fused at a higher level than the branch device, which in this case would be the DL105 unit. This double fusing technique is required when operating from direct current. The recommended fuse size for the branch circuit to the DL105 is 0.5A (for example, a Littlefuse 312.500 or equivalent).

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 DL105 input power terminal can accommodate one 14 AWG or two 16 AWG wires. Each terminal block junction and/or connection creates a voltage drop, so try to keep the number of connections to a minimum. In general, when using 12/24 VDC input power, observe the same double fusing techniques that are used with 125 VDC input power. The recommended fuse size for the branch circuit to the DL105 is 1A (for example, a Littlefuse 312.001 or equivalent).

External Power Source

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

Part Numbers	F1-130AR, F1-130DR, F1-130AD, F1-130DD, F1-130AA, F1-130DA	F1-130DR-D, F1-130DD-D
Input Voltage Range	85–132 VAC (110 nominal) 170–264 VAC (220 nominal), 100 – 264 VDC (125 nominal)	10 – 30 VDC (12 to 24VDC) with less than 10% ripple
Maximum Inrush Current	12 A, <1/2 mS	12 A, <1/2 mS
Maximum Power	30 VA (for AC power) 30 W (for DC power)	10W (0.3A @ 30VDC)
Voltage Withstand (dielectric)	1 minute @ 1500 VAC between primary, secondary, field ground	
Insulation Resistance	> 10 MΩ at 500 VDC	
Auxiliary 24 VDC Output	21.6–26.4 VDC Ripple less than 200 mV p-p 500 mA max, isolated.	None



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

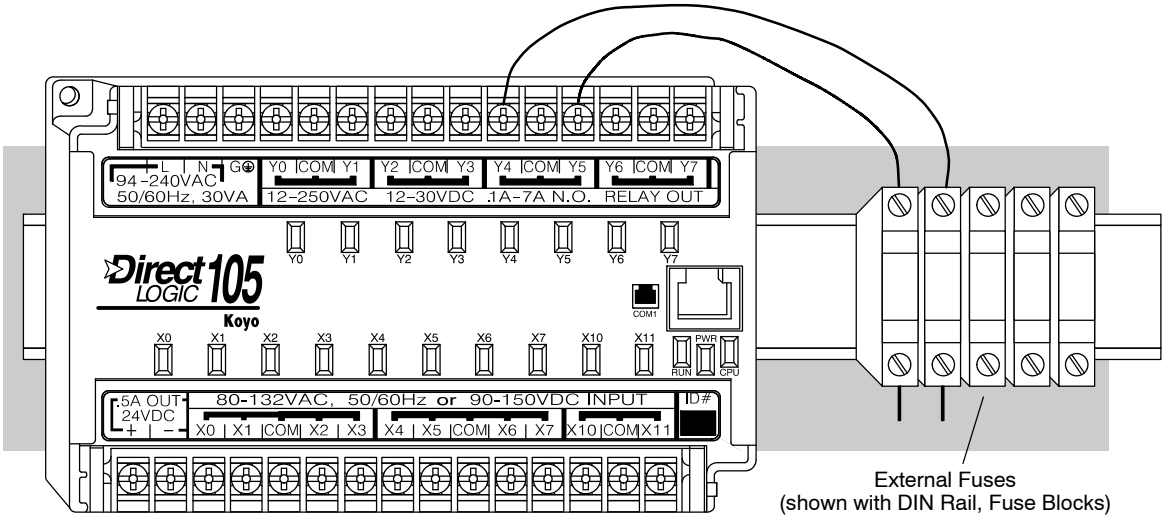
Planning the Wiring Routes

The following guidelines provide general information on how to wire the I/O connections to DL105 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 DL105 PLC can accept one 14 AWG wire or two 16 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.

Fuse Protection for Input and Output Circuits

Input and Output circuits on DL105 Micro PLCs do not have internal fuses. However, the +24V Auxiliary Supply is current-limited. In order to protect your Micro PLC, we suggest you add external fuses to your I/O wiring. A fast-blow fuse, with a lower current rating than the I/O bank's common current rating can be wired to each common. Or, a fuse with a rating of slightly less than the maximum current per output point can be added to each output. Refer to the Micro PLC specification sheets further in this chapter to find the maximum current per output point or per output common. Adding the external fuse does not guarantee the prevention of Micro PLC damage, but it will provide added protection.



I/O Point Numbering

All DL105 Micro PLCs have a fixed I/O configuration. It follows the same octal numbering system used on other ***Direct***LOGIC family PLCs, starting at X0 and Y0. The letter X is always used to indicate inputs and the letter Y is always used for outputs.

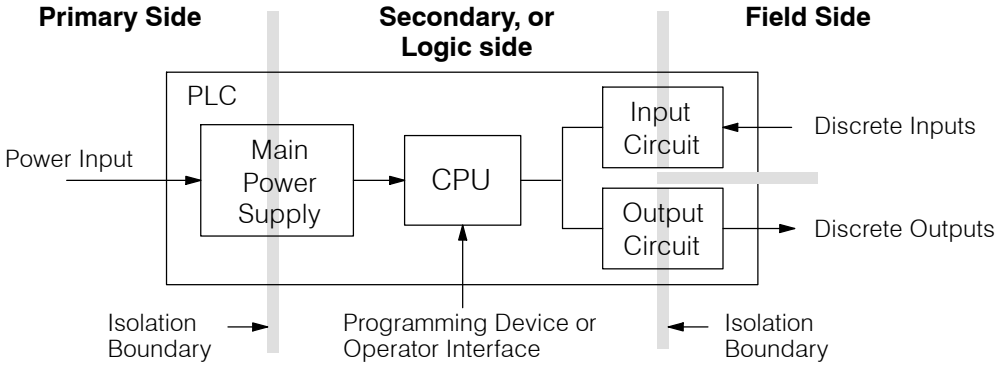
The I/O numbering always starts at zero and does not include the digits 8 or 9. The addresses are typically assigned in groups of 8 or 16, depending on the number of points in an I/O group. For the DL105 the ten inputs use reference numbers X0 – X7 and X10 – X11. The eight output points use references Y0 – Y7.

System Wiring Strategies

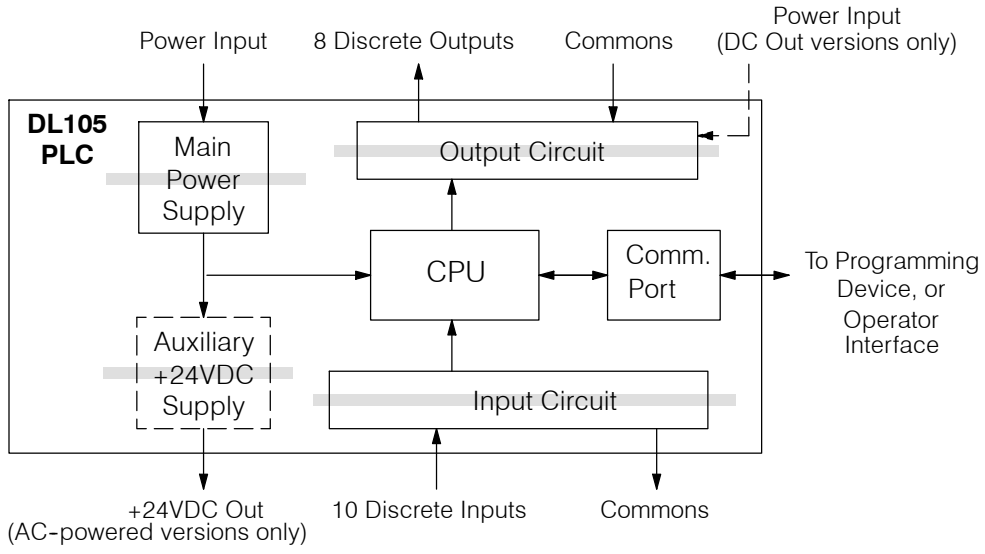
The DL105 Micro PLC is very flexible and will work in many different wiring configurations. By studying this section before actual installation, you can probably find the best wiring strategy for your application . This will help to lower system cost, wiring errors, and avoid safety problems.

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 transformer in the power supply provides magnetic isolation between the primary and secondary sides. Opto-couplers provide optical isolation in Input and Output circuits. This isolates logic circuitry from the field side, where factory machinery connects. Note that the discrete inputs are isolated from the discrete outputs, because each is isolated from the logic side. Isolation boundaries protect the operator interface (and the operator) from power input faults or field wiring faults. *When wiring a PLC, it is extremely important to avoid making external connections that connect logic side circuits to any other.*



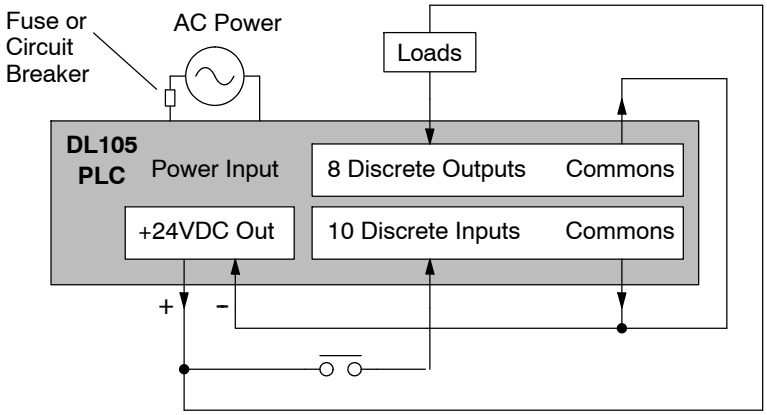
The next figure shows the internal layout of DL105 PLCs, as viewed from the front panel. In addition to the basic circuits covered above, it includes an auxiliary +24VDC power supply with its own isolation boundary. Since the supply output is isolated from the other three circuits, it can power input and/or output circuits!



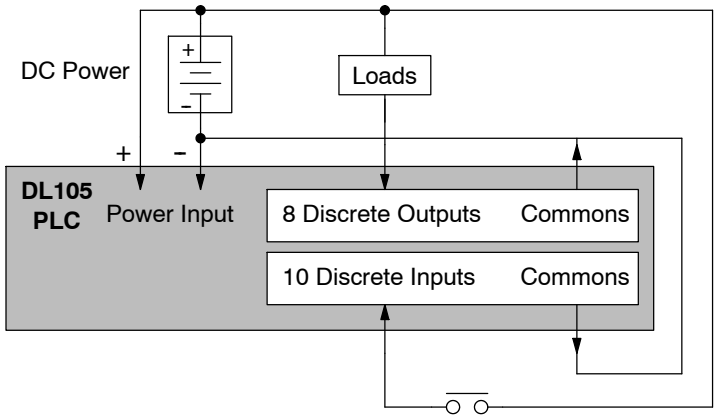
In many cases, using the built-in auxiliary +24VDC supply can result in a cost savings for your control system. It can power combined loads up to 500 mA, which is enough to eliminate the need for an additional power supply in some applications. If you are the system designer for your application, you may be able to select and design in field devices which can use the +24VDC auxiliary supply.

Powering I/O Circuits with the Auxiliary Supply

All AC-powered DL105 Micro PLCs feature the internal auxiliary supply. If input devices AND output loads need +24VDC power, the auxiliary supply can power both circuits as shown in the following diagram.



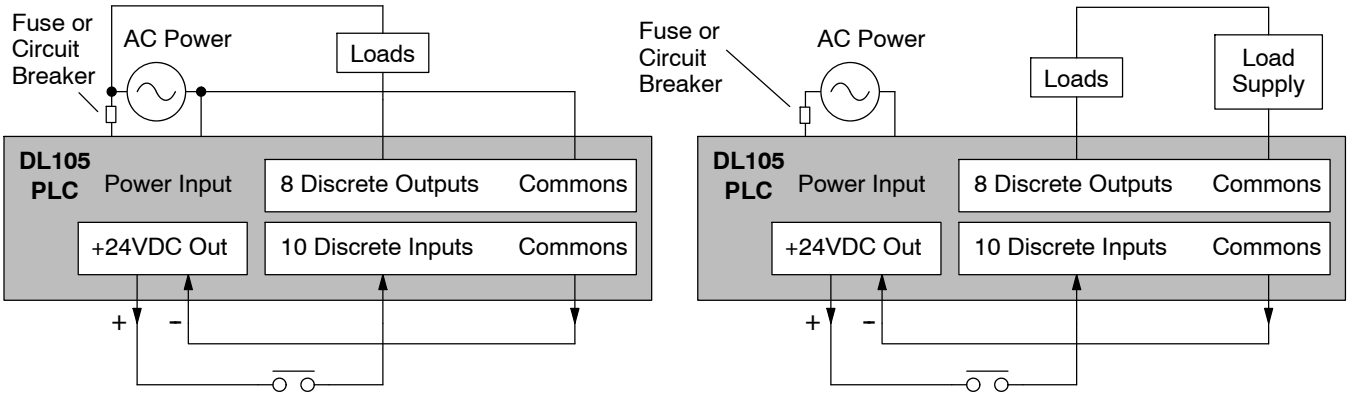
DC-powered DL105 Micro PLCs are designed for application environments in which low-voltage DC power is more readily available than AC. These include a wide range of battery–powered applications, such as remotely-located control, in vehicles, portable machines, etc. For this application type, all input devices and output loads typically use the same DC power source. The F1-130DR-D and F1-130DD-D are compatible with either +12VDC or +24VDC systems. Typical wiring for DC-powered applications is shown in the following diagram.



Powering I/O Circuits Using Separate Supplies

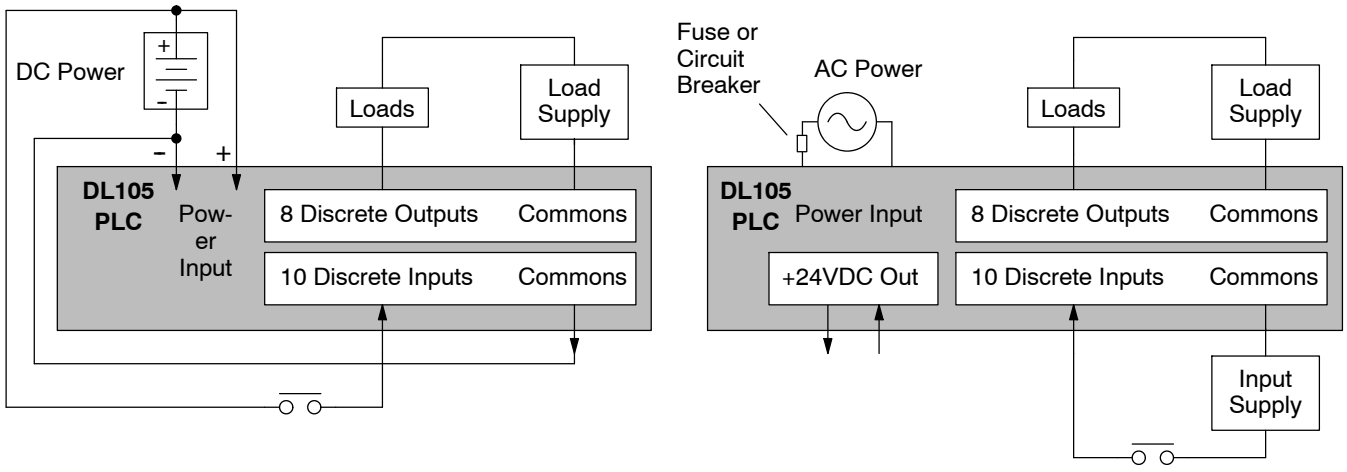
In some applications it will be necessary to power the input devices from one power source, and to power output loads from another source. Loads often require high-energy AC power, while input sensors use low-energy DC. If a machine operator is likely to come in close contact with input wiring, then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

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



Some applications will use the Micro PLC power source to also power the input circuit. This typically occurs on a DC-powered DL105, as shown in the drawing below to the left. The inputs share the PLC power source supply, while the outputs have their own separate supply.

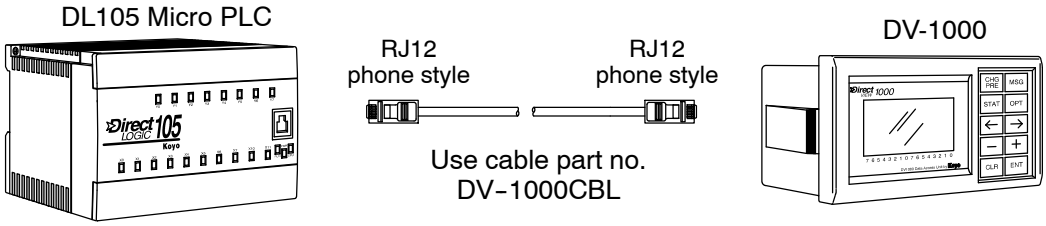
A worst-case scenario, from a cost and complexity view-point, is an application which requires separate power sources for the PLC, input devices, and output loads. The example wiring diagram below on the right shows how this can work, but also that the auxiliary supply out is an unused resource. For these reasons, you'll probably want to avoid this situation if possible.



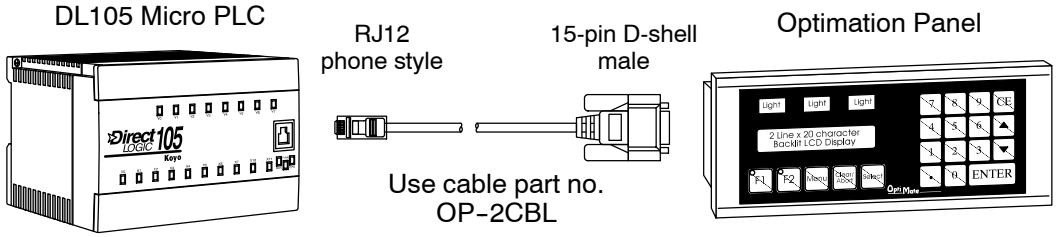
Connecting Operator Interface Devices

Operator interfaces require data and power connections. Operator interfaces with a large CRT usually require separate AC power. However, small operator interface devices like the DV-1000 Data Access Unit, some C–more Micro–Graphic panels and some Optimation panels may be powered directly from the DL105 Micro PLC.

Connect the DV-1000 to the DL105 Micro PLC COM1 port using the cable shown below. A single cable contains transmit/receive data wires and +5V power.

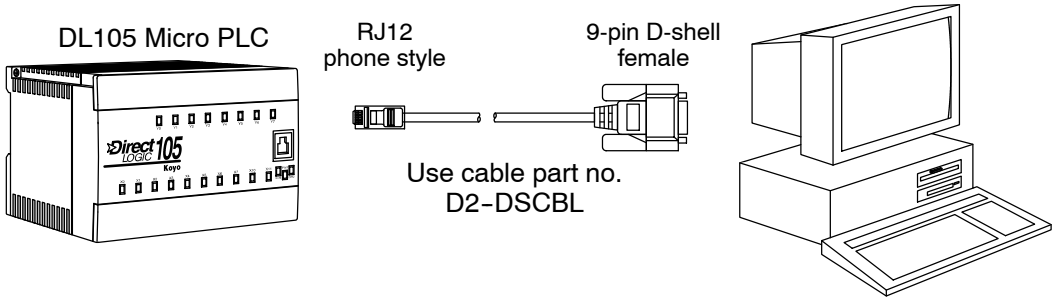


Optimation operator interface panels require separate power and communications connections. Connect the DL105 COM1 port to the 15-pin D-shell connector on the rear of the Optimation panel using the cable shown below. Optimation panels require 8–30VDC power, so use separate wiring to connect the +24VDC supply output on AC-powered DL105 PLCs. Use external +24VDC power for DC-powered DL105s.

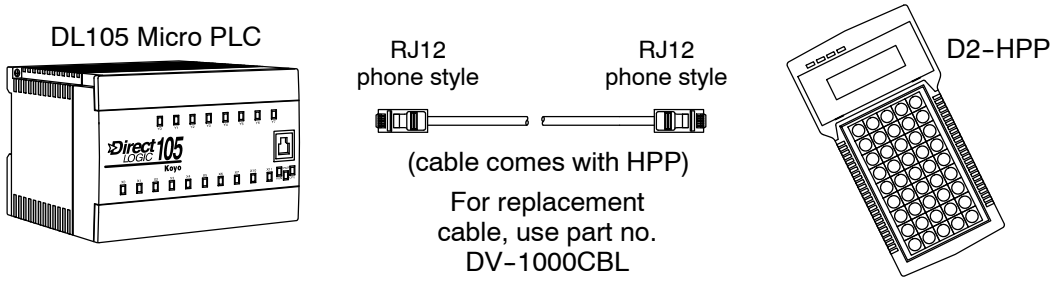


Connecting Programming Devices

DL105 Micro PLCs can be programmed with either a handheld programmer or with ***Direct***SOFT32 on a PC. Connect the DL105 to a PC using the cable shown below.



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



Sinking / Sourcing Concepts

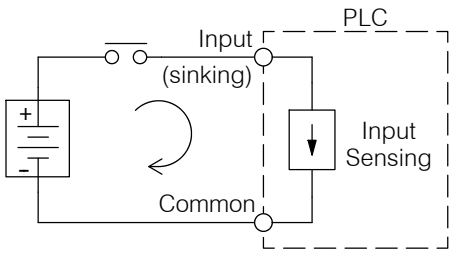
Before going further in our study of wiring strategies, we must have a solid understanding of "*sinking*" and "*sourcing*" concepts. Use of these terms occurs frequently in input or output circuit discussions. It is the goal of this section to make these concepts easy to understand, further ensuring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

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

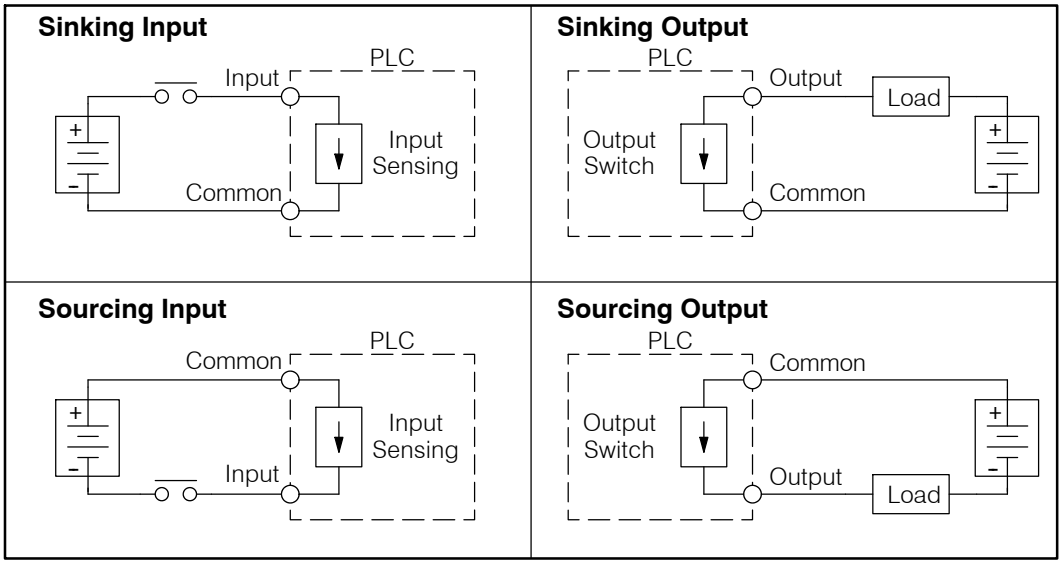
Sourcing = Path to supply **source (+)**

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 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.

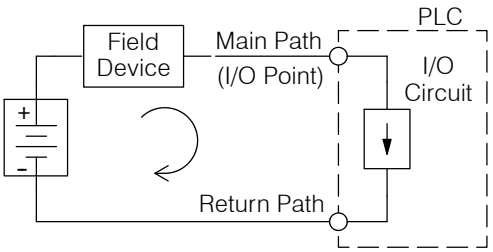


By applying the circuit principle above to the four possible combinations of input/output sinking/sourcing types, we have the four circuits as shown below. DL105 Micro PLCs provide all except the sourcing output I/O circuit types.

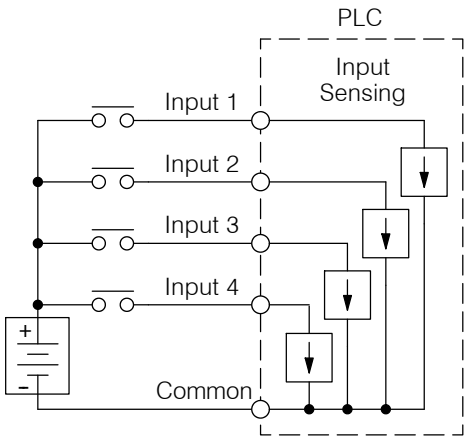


I/O "Common" Terminal Concepts

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.

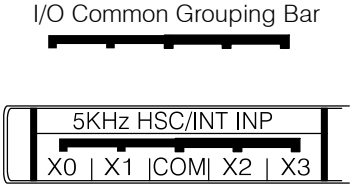


If we had unlimited space and budget for I/O terminals, then every I/O point could have two dedicated terminals just as the figure above shows. However, providing this level of flexibility is not practical or even necessary for most applications. So, most Input or Output point groups on PLCs share the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

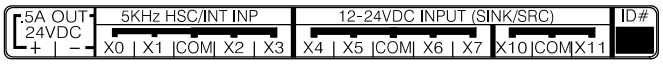


Note: In the circuit above, the current in the common path is 4 times any channel's input current when all inputs are energized. This is especially important in output circuits, where heavier gauge wire is sometimes necessary on commons.

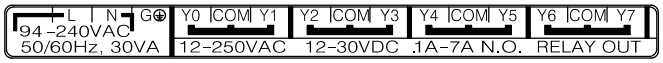
Most DL105 input and output circuits are grouped into banks that share a common return path. The best indication of I/O common grouping is on the wiring label. The I/O common grouping bar, labeled at the right, occurs in the section of wiring label below it. It indicates X0, X1, X2, and X3 share the common terminal located between X1 and X2.



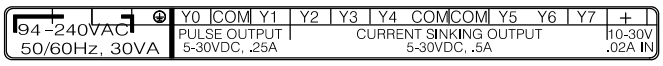
The following complete label shows two banks of four inputs and one bank of two.



The following label for relay outputs shows four banks of two output points each.



The last label below for DC outputs has no common grouping bar. In this unique case, all eight outputs share the same electrical common. The common is available on three terminals, so there is a physical place to connect each point's common wire.

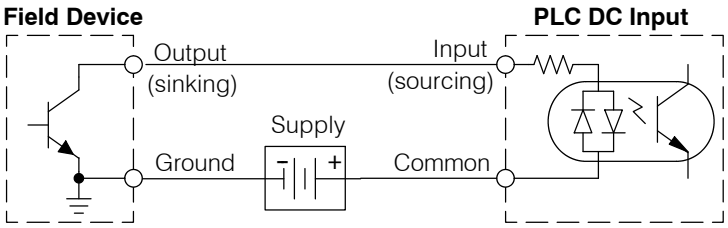


Connecting DC I/O to "Solid State" Field Devices

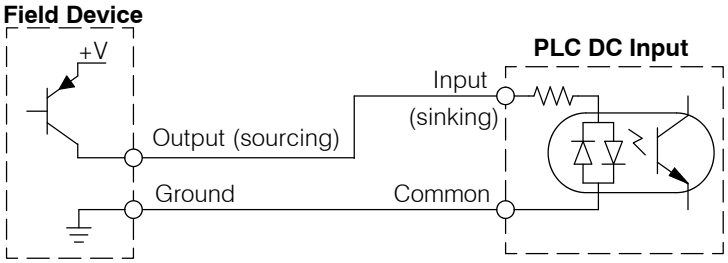
In the previous section on Sourcing and Sinking concepts, we explained that DC I/O circuits sometimes will only allow current to flow one way. This is also true for many of the field devices which have solid-state (transistor) interfaces. In other words, field devices can also be sourcing or sinking. *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 DL105'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 +24 auxiliary supply or another supply (+12 VDC or +24VDC), as long as the input specifications are met.



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

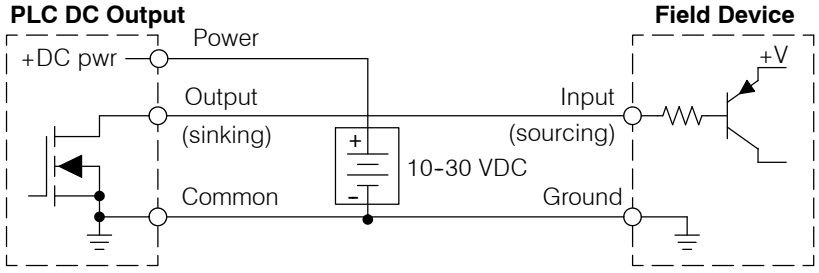


Solid State Output Loads

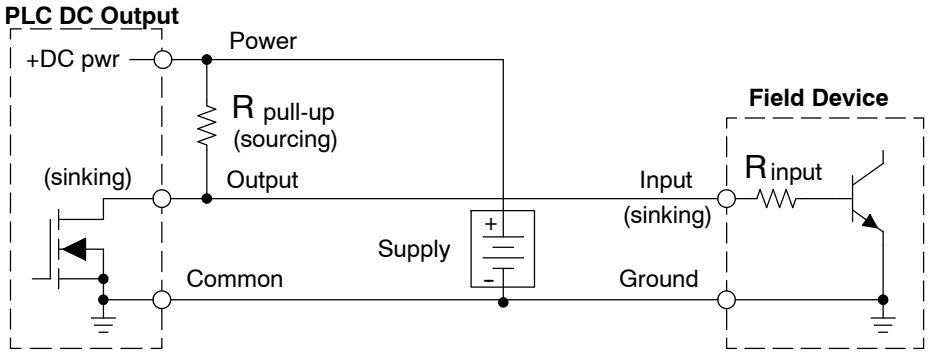
Sometimes an application requires connecting a PLC output point to a solid state input on a device. This type of connection is usually made to carry a low-level signal, not to send DC power to an actuator.

The DL105'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 eight outputs have the same electrical common, even though there are three common terminal screws. Finally, recall that the DC output circuit requires power (10 – 30 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 $R_{pull\text{-}up}$ from the output to the DC output circuit power input.



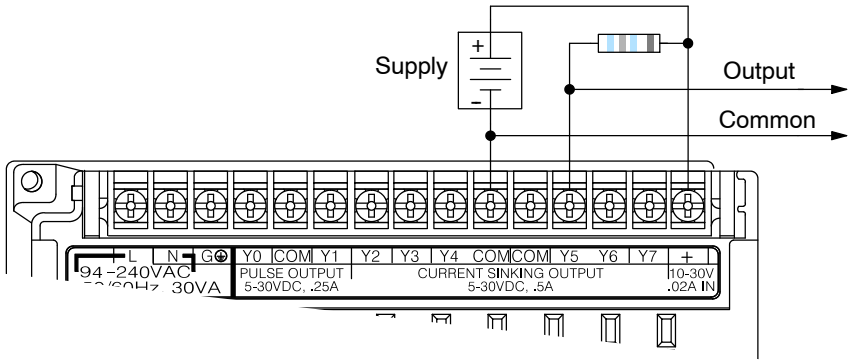
NOTE 1: DO NOT attempt to drive a heavy load (>25 mA) with this pull-up method.
NOTE 2: Using the pull-up resistor to implement a sourcing output has the effect of inverting the output point logic. In other words, the field device input is energized when the PLC output is OFF, from a ladder logic point-of-view.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

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

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

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

The drawing below shows the actual wiring of the DL105 Micro PLC to the supply and pull-up resistor.



Relay Output Wiring Methods

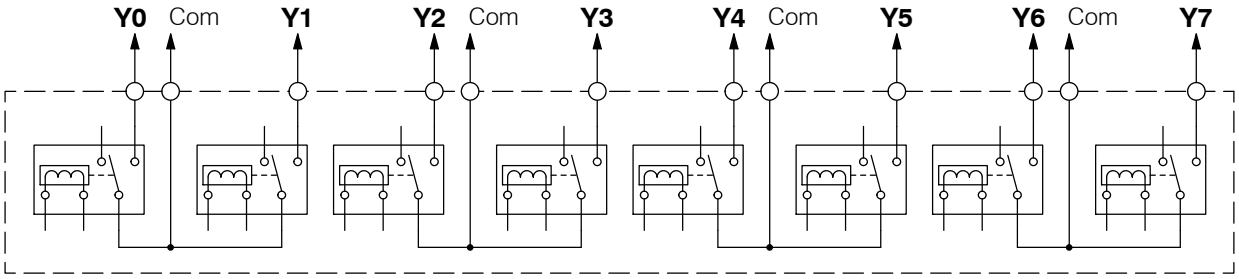
The F1-130AR, F1-130DR/F1-130DR-CE, and F1-130DR-D models feature relay outputs. Relays are best for the following applications:

- Loads that require higher currents than the solid-state DL105 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:

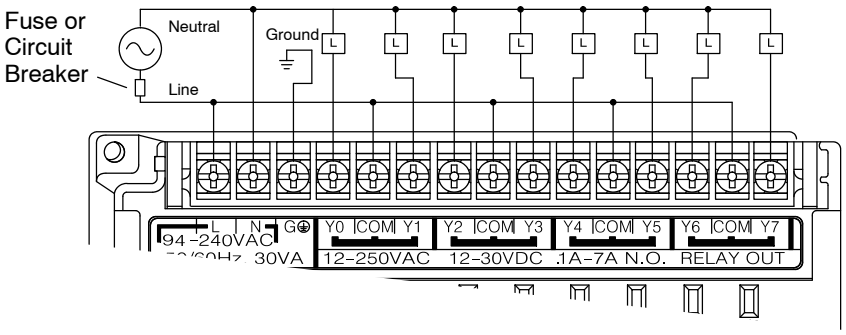
- Loads that require currents under 10 mA
- Loads which must be switched at high speed and duty cycle

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 eight normally-open SPST relays available. They are organized into four pairs with individual commons. The figure below shows the relays and the internal wiring of the PLC. Note that each pair is isolated from the other three relay pairs.

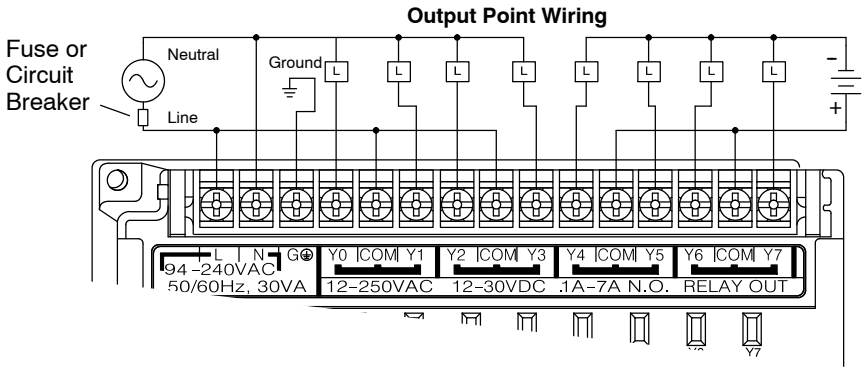


In the circuit below, all loads use the same AC power supply which powers the DL105 PLC. In this example, all commons are connected together.

Output Point Wiring



In the circuit on the following page, loads for Y0 – Y3 use the same AC power supply which powers the DL105 PLC. Loads for Y4 – Y7 use a separate DC supply. In this example, the commons are separated according to which supply powers the associated load.



Surge Suppresion For Inductive Loads

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

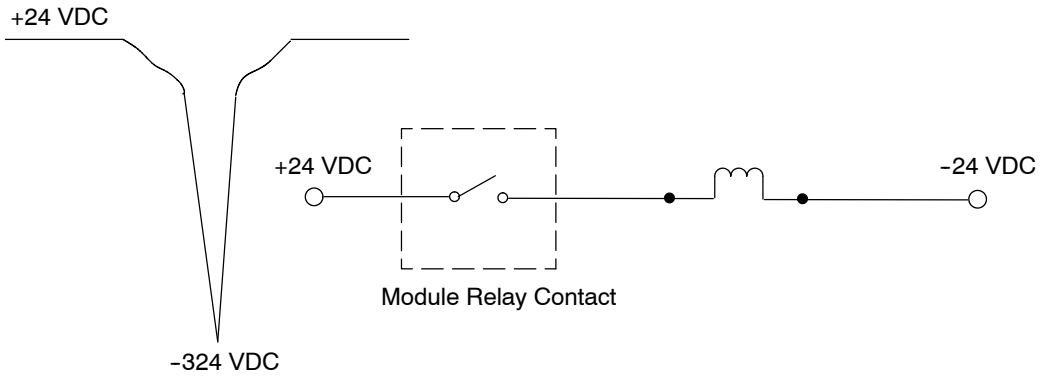
When switching a DC-supplied inductive load the full supply voltage is always present when the relay contact opens (or "bounces"). When switching an AC-supplied inductive load there is one chance in 60 (60 Hz) or 50 (50 Hz) that the relay contact will open (or "bounce") when the AC sine wave is zero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

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

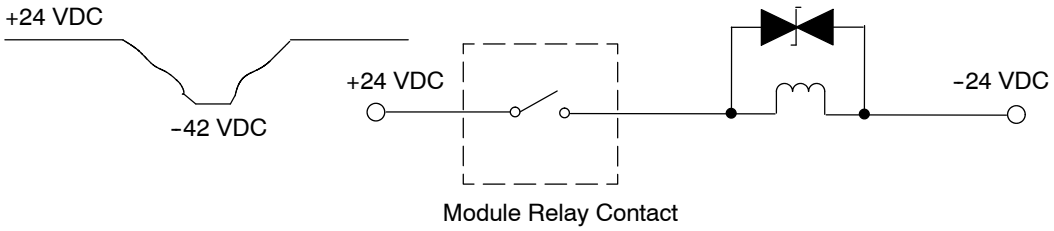
Transient Voltage Suppressors (TVS or transorb) provide the best surge and transient suppression of AC and DC powered coils, providing the fastest response with the smallest overshoot.

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

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



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



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

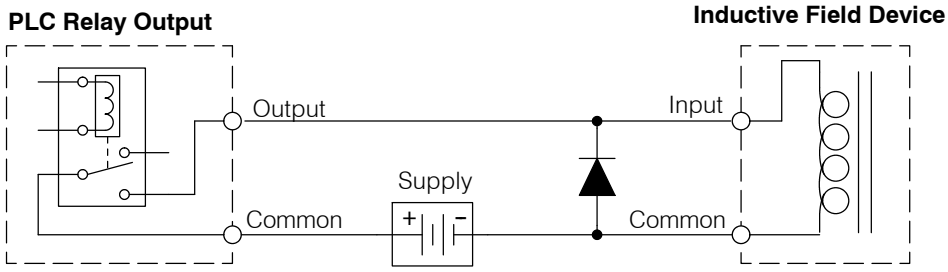
Surge Suppressors			
Vendor / Catalog	**Type (TVS, MOV, Diode)**	**Inductive Load Voltage**	**Part Number**
AutomationDirect Transient Voltage Suppressors www.automationdirect.com	TVS TVS	110/120 VAC 24 VDC	ZL-TD8-120 ZL-TD8-24
General Instrument Transient Voltage Suppressors and LiteOn Diodes: from-Digi-Key Catalog: www.digikey.com; Phone: 1-800-344-4539	TVS, MOV TVS, MOV TVS Diode	110/120 VAC 220/240 VAC 12/24 VDC or VAC 12/24 VDC or VAC	Contact Digi-Key, Corp. catalog or website

Prolonging Relay Contact Life

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

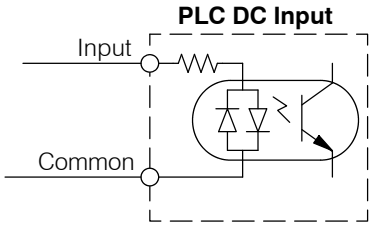
For inductive loads in DC circuits we recommend using a suppression diode as shown in the following diagram (DO NOT use this circuit with an AC power supply). When the load is energized the diode is reverse-biased (high impedance). When the load is turned off, energy stored in its coil is released in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

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

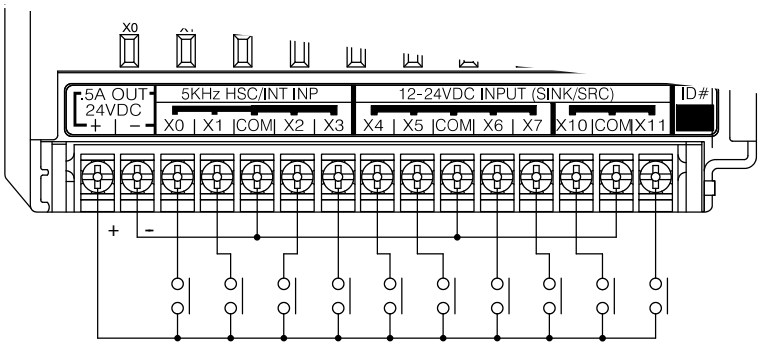


DC Input Wiring Methods

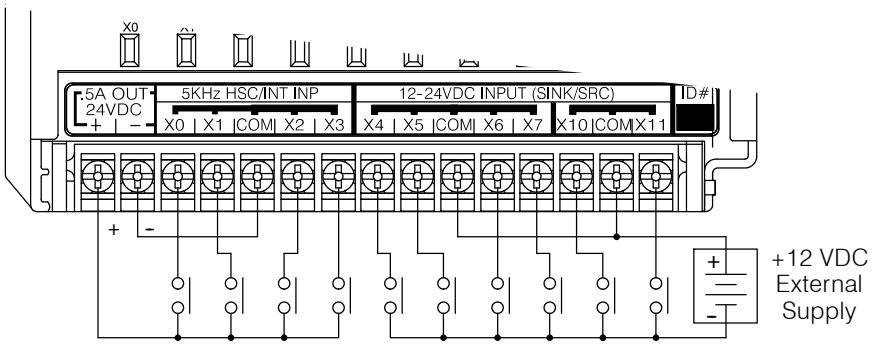
DL105 Micro PLCs with DC inputs are particularly flexible because they can be either sinking or sourcing. The dual diodes (shown to the right) allow current to flow in either direction. The inputs accept either 10 – 26.4 VDC or 21.6 – 26.4 VAC. That's right, either AC or DC voltages will work. The target applications are +12 VDC, +24 VDC, and 24 VAC. You can actually wire part of the inputs as DC sinking, others as DC sourcing, and the rest as AC!



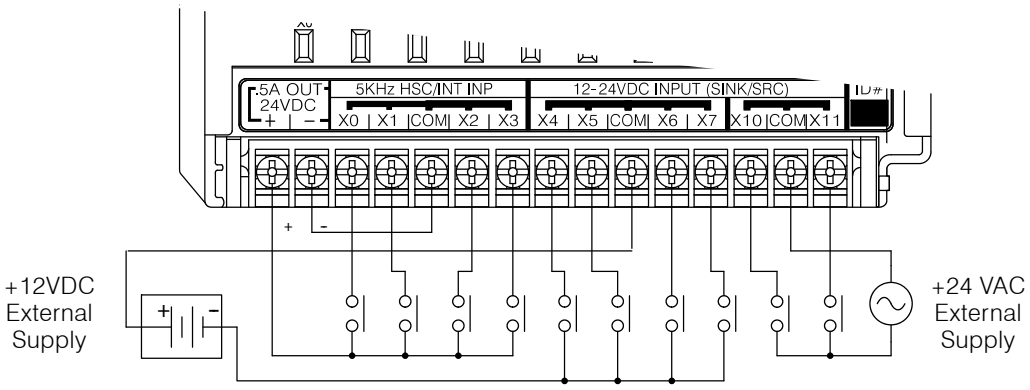
In the first and simplest example below, all commons are connected together and all inputs are sinking.



In the next example, the first four inputs are sinking, and the last six are sourcing.



In the last example, four inputs are sinking DC, four are sourcing DC, and two are AC.



DC Output Wiring Methods

DL105 DC output circuits are high-performance MOSFET switches with low on-resistance and fast switching times. Please note the following characteristics which are unique to the DC output type:

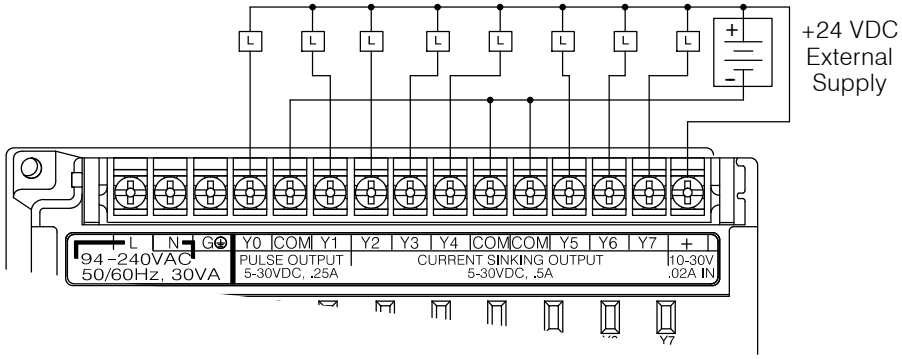
- There is only one electrical common for all eight outputs, even though there are three common terminals. All eight outputs belong to one bank.
- The output switches are current-sinking only. However, you can still use different DC voltages from one load to another.
- The output circuit inside the PLC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



NOTE: Always connect all three common terminals together at the connector with short wires (do not leave some common terminals unconnected).This provides three connections to share the load return current, enhancing reliability.

In the example below, all eight outputs share a common supply. It may be external as shown, or they may use the auxiliary +24VDC supply when available.

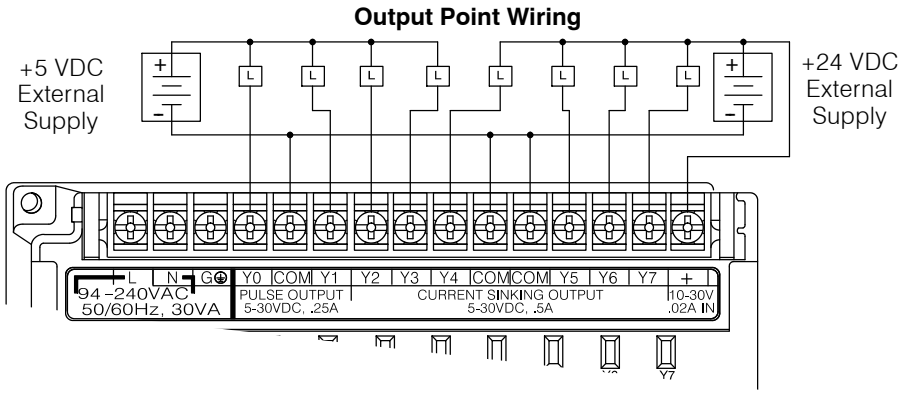
Output Point Wiring



In the next example below, the outputs have “split” supplies. The first four outputs are using a +5 VDC supply, and the last four are using a +24 VDC supply. However, you can split the outputs among any number of supplies, as long as:

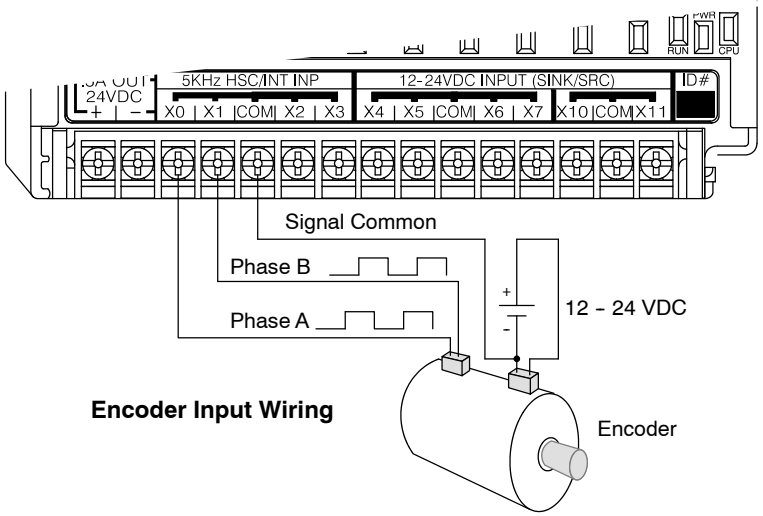
- all supply voltages are within the specified range
- all output points are wired as sinking
- all source (–) terminals are connected together

Output Point Wiring



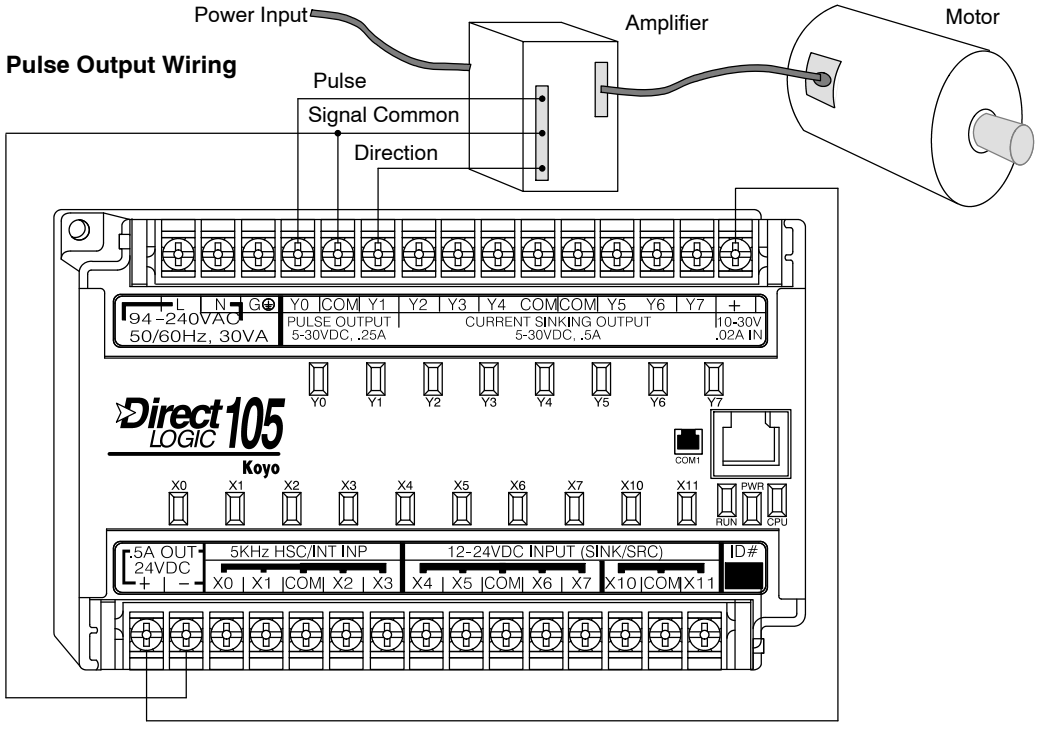
High-Speed I/O Wiring Methods

DL105 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 4 discusses the programming options for HSIO. While the HSIO circuit has six modes, we show wiring diagrams for two of the most popular modes in this chapter. The high-speed input interfaces to points X0 – X3. Properly configured, the DL105 can count quadrature pulses at up to 5 kHz from an incremental encoder as shown below.



Encoder Input Wiring

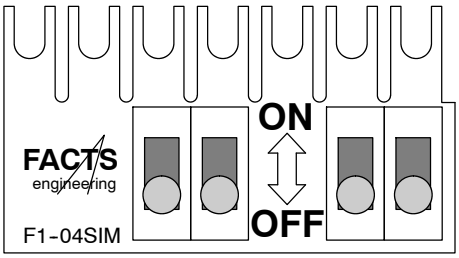
DL105 versions with DC type output points can use the High Speed I/O Pulse Output feature. It can generate high-speed pulses for specialized control such as stepper motor / intelligent drive systems. Outputs Y0 and Y1 can generate pulse and direction signals, or they can generate CCW and CW pulse signals respectively. See Chapter 3 on high-speed input and pulse output options.



Pulse Output Wiring

F1-04SIM Input Simulator Wiring

The F1-04SIM Input Simulator, shown to the right, provides four switches for inputs X0 through X3. The simulator is useful during program development or for debug purposes. It works by using the +24VDC auxiliary supply output, routing the voltage through the switches and into the inputs.

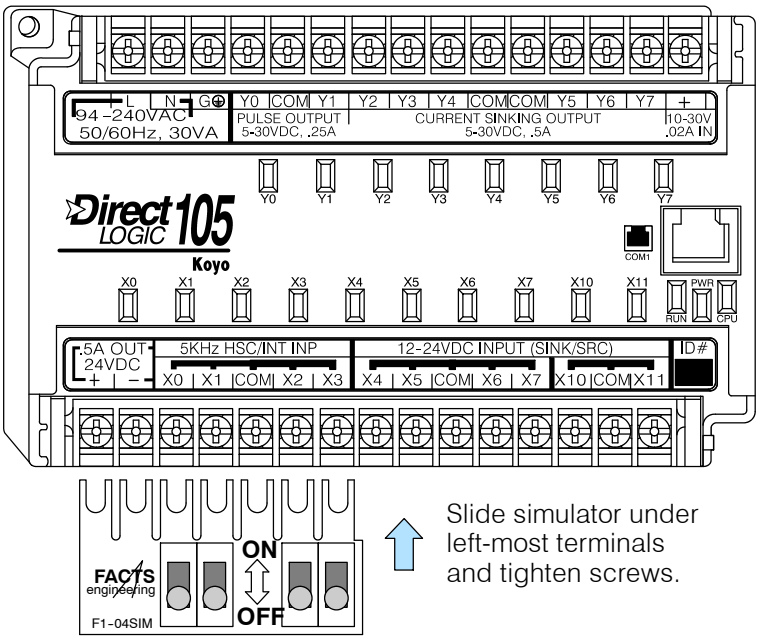


In use, the simulator can quickly provide test inputs to your ladder program. The status of outputs is observable on the front panel LEDs, even without wiring the outputs to any loads.

The Simulator works on all DC input versions of the DL105. DC-powered versions need two wires from the power input to connect to the two left-most terminals on the simulator (wiring shown below), since DC-powered units do not generate +24VDC auxiliary output. Polarity does not matter, since the inputs can be sinking or sourcing.

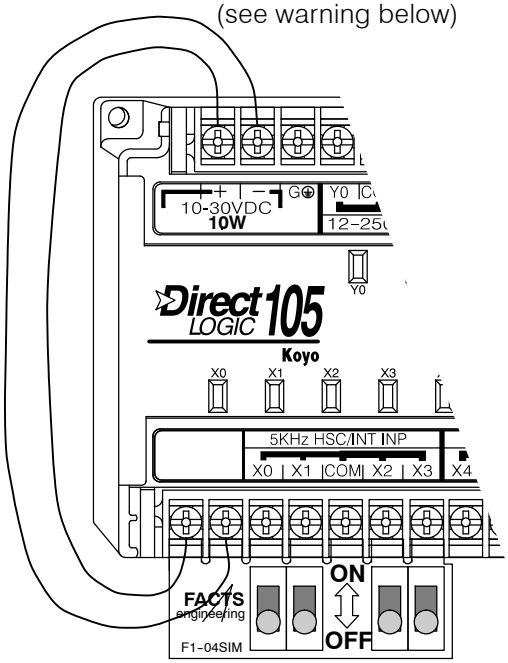
NOTE: The Input Simulator will not work on DL105 micros with AC type inputs. The +24 VDC auxiliary supply voltage is less than the required input threshold.

F1-130DR/F1-130DR-CE, F1-130DD/F1-130DD-CE, F1-130DA



Slide simulator under left-most terminals and tighten screws.

F1-130DR-D, F1-130DD-D only (see warning below)



NOTE: Never attempt to install more than one simulator on one DL105 PLC.

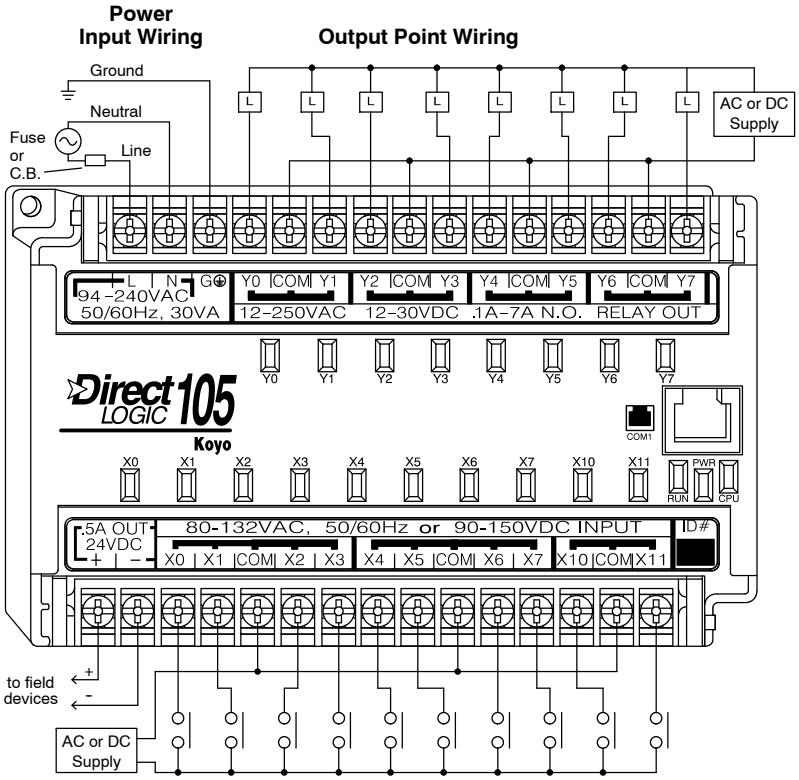
WARNING: DO NOT use the two wires as shown above on AC-powered DL105 PLCs. Doing so will permanently damage the Micro PLC and may result in electrical shock due to the exposed circuit board of the input simulator.

Wiring Diagrams and Specifications

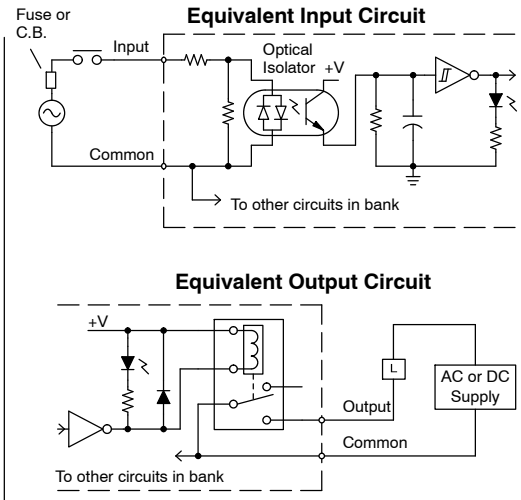
The remainder of this chapter dedicates two to three pages to each of the eight versions of DL105 Micro PLCs. Each section contains a basic wiring diagram, equivalent I/O circuits, and specification tables. Please refer to the section which describes the particular DL105 version used in your application.

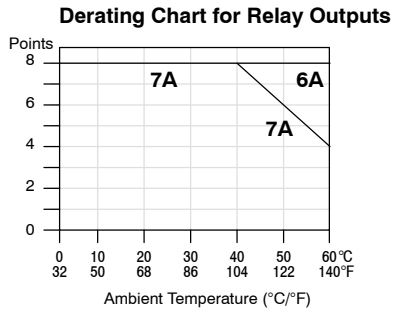
F1-130AR I/O Wiring Diagram

The F1-130AR Micro PLC features ten AC inputs and eight relay contact outputs. The following diagram shows a typical field wiring example. The AC external power connection uses three terminals at the top left as shown.



Input Point Wiring





The ten AC input channels use terminals on the bottom connector. This input type also works for high-voltage DC signals. Inputs are organized into two banks of four, plus one bank of two. Each bank has a common terminal. In the case of DC input signals, the input may be wired in as either the sourcing or sinking type. 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 eight relay output channels use terminals on the top connector. Outputs are organized into four banks of two 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.

Auxiliary +24V Power Supply

The F1-130AR has a +24V supply output to power external devices. The output is rated at 0.5 Amperes, and includes short-circuit protection and full isolation from internal CPU circuitry. These features make it ideal for powering sensors, solenoids, and other field devices. In fact, it can be used as the DC supply for loads in the relay output circuits. Be sure the combined load currents do not exceed 0.5 A. Note that on the F1-130AR, the +24V auxiliary output is not high enough to power its input circuits (input ON threshold is 90VDC).

F1-130AR General Specifications

External Power Requirements	100 - 240 +10% -15%
Communication Port	K-Sequence, 9600 baud, 8 data bits, odd parity
Programming cable type	D2-DSCBL
Internal Field Supply Ratings	+24VDC , 0.5A maximum, isolated
Operating Temperature	32 to 140° F (0 to 60° 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 AWG14 or two AWG16, AWG24 minimum

AC Input Specifications X0 - X7, X10 - X11

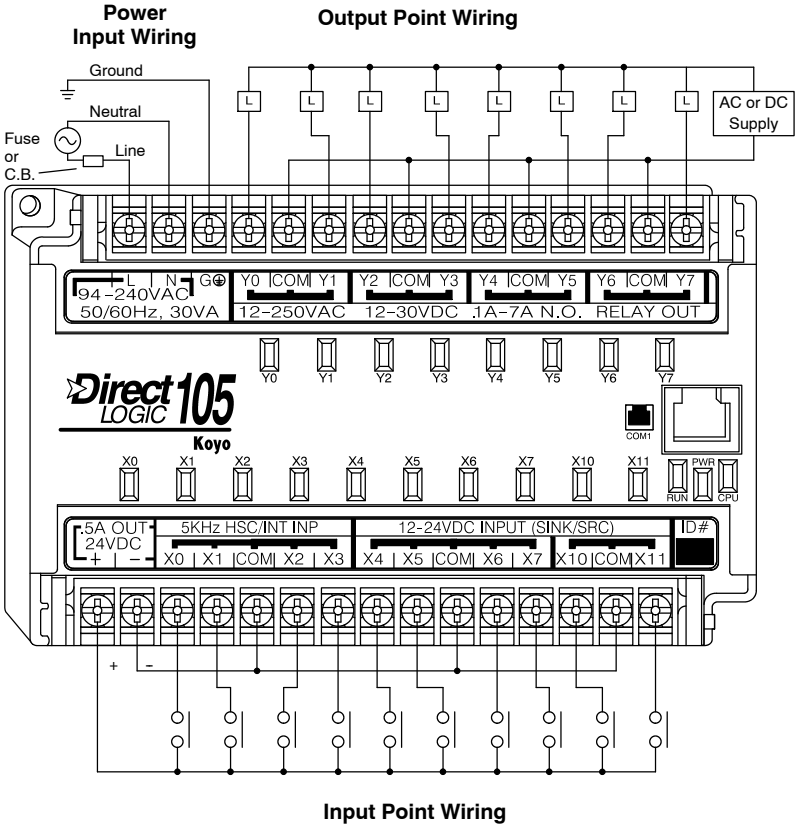
Input Voltage Range for ON condition	80 - 132 VAC, or 90 - 150 VDC
Input Current	6 mA @ 132 VAC 6.8 mA @ 150 VDC
Maximum Voltage	132 VAC, or 150 VDC
ON Current/Voltage	>4 mA @ 80 VAC, or 90 VDC
OFF Current/Voltage	<2 mA @ 45 VAC, or 60 VDC
OFF to ON Response	< 8 mS
ON to OFF Response	<15 mS
Status Indicators	Logic Side
Commons	4 channels / common x 2 banks, 2 channels / common x 1 bank

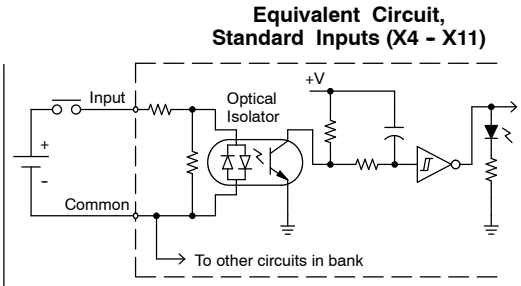
Relay Output Specifications Y0 - Y7

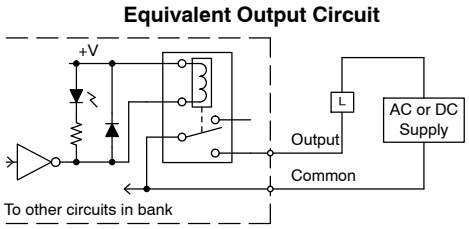
Operating Voltage	12 - 250 VAC, 12 - 30 VDC @ 7A, 30 - 150 VDC @ 0.5A, resistive
Output Current	7A / point (subject to derating) 14A / common
Maximum Motor Load	1/3 HP
Maximum Voltage	265 VAC, 150 VDC
Minimum Off Resistance	100 meg ohms @ 500 VDC
Smallest Recommended Load	10 mA
OFF to ON Response	15 mS
ON to OFF Response	5 mS
Status Indicators	Logic Side
Commons	2 channels / common x 4 banks
Fuses	None (external recommended)

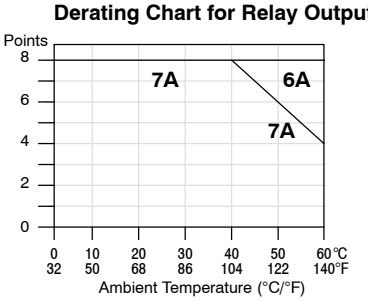
F1-130DR/ F1-130DR-CE I/O Wiring Diagram

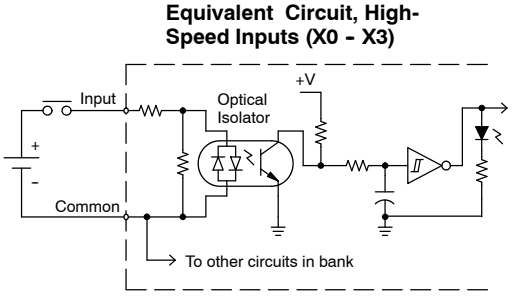
These micro PLCs feature ten DC inputs and eight relay contact outputs. The following diagram shows a typical field wiring example. The AC external power connection uses three terminals at the top left as shown.











The ten DC input channels use terminals on the bottom connector. Inputs are organized into two banks of four, plus one bank of two. 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 is shown above, and the high-speed input circuit is shown to the right.

The eight output channels use terminals on the top connector. Outputs are organized into four banks of two 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.

Auxiliary +24V Power Supply

These versions have a +24V supply output to power external devices. The output is rated at 0.5 Amperes, and includes short-circuit protection and full isolation from internal CPU circuitry. These features make it ideal for powering sensors, solenoids,

and other field devices. In fact, it can be used as the DC supply for switches or sensors in the input circuit, or for loads in the relay output circuits. Be sure the combined load currents do not exceed 0.5 A.

F1-130DR/ F1-130DR-CE General Specifications

External Power Requirements	100 - 240 +10% -15%
Communication Port	K-Sequence, 9600 baud, 8 data bits, odd parity
Programming cable type	D2-DSCBL
Internal Field Supply Ratings	+24VDC , 0.5A maximum, isolated
Operating Temperature	32 to 140° F (0 to 60° 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 AWG14 or two AWG16, AWG24 minimum

DC Input Specifications

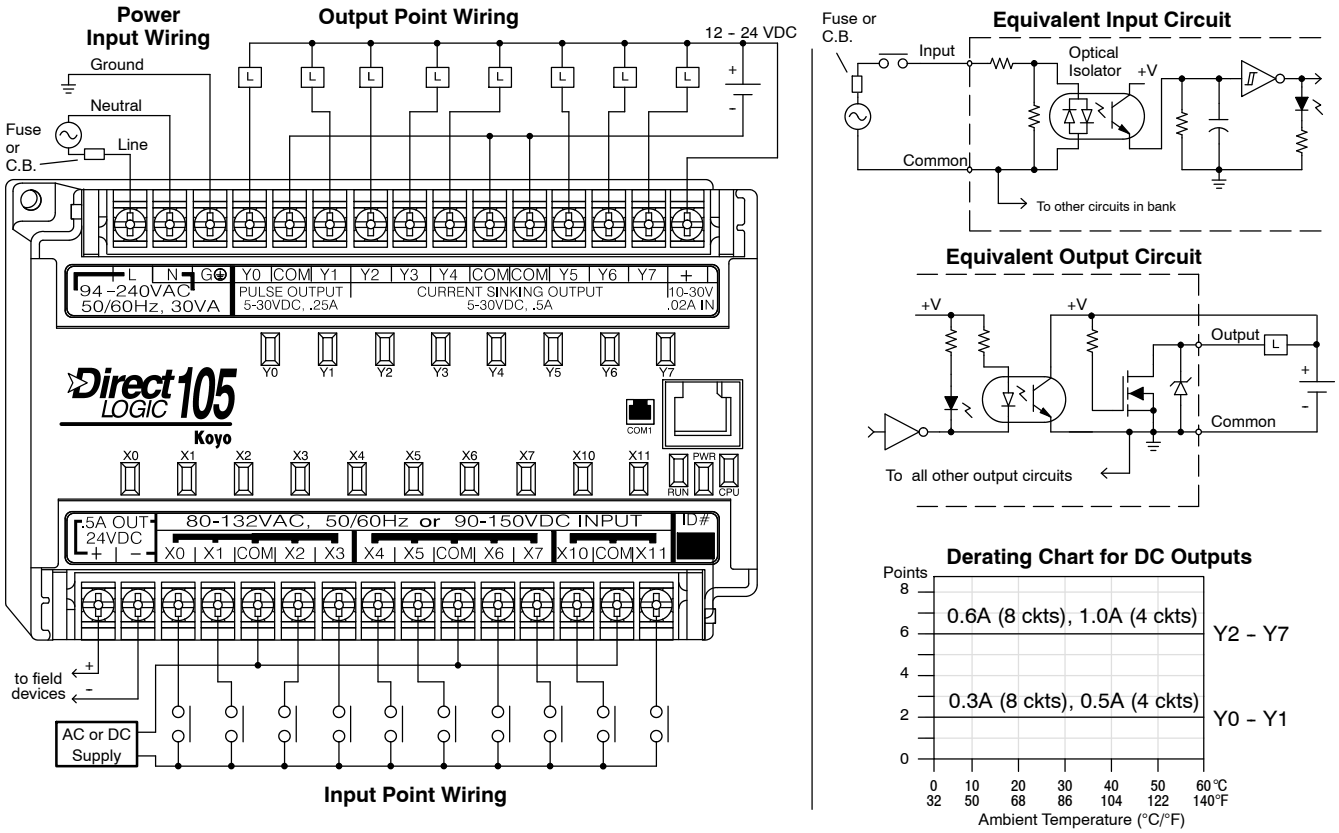
Parameter	High-Speed Inputs, X0 - X3	Standard DC Inputs X4 - X11
Input Voltage Range	10 - 26.4 VDC	10 - 26.4 VDC or 21.6 - 26.4 VAC
Maximum Voltage	30 VDC (5 kHz maximum frequency)	30 VDC
Minimum Pulse Width	100 μs	N/A
ON Voltage Level	> 9.0 VDC	> 9.0 VDC
OFF Voltage Level	< 2.0 VDC	< 2.0 VDC
Input Impedance	2.8 kΩ @ 12 - 24 VDC	2.8 kΩ @ 12 - 24 VDC
Minimum ON Current	>3 mA	>3 mA
Maximum OFF Current	< 0.5 mA	<0.5 mA
OFF to ON Response	<50 μs	2 - 8 mS, 4 mS typical
ON to OFF Response	< 50 μs	2 - 8 mS, 4 mS typical
Status Indicators	Logic side	Logic side
Commons	4 channels / common x 1 bank	4 channels / common x 1 bank, 2 channels / common x 1 bank

Relay Output Specifications

Operating Voltage	12 - 250 VAC, 12 - 30 VDC @ 7A, 30 - 150 VDC @ 0.5A, resistive
Output Current	7A / point (subject to derating) 14A / common
Maximum Motor Load	1/3 HP
Maximum Voltage	265 VAC, 30 VDC
Minimum Off Resistance	100 meg ohms @ 500 VDC
Smallest Recommended Load	10 mA
OFF to ON Response	15 mS
ON to OFF Response	5 mS
Status Indicators	Logic Side
Commons	2 channels / common x 4 banks
Fuses	None (external recommended)

F1-130AD I/O Wiring Diagram

The F1-130AD Micro PLC features ten AC inputs and eight DC outputs. The following diagram shows a typical field wiring example. The AC external power connection uses three terminals at the top left as shown.



The ten AC input channels use terminals on the bottom connector. This input type also works for high-voltage DC signals. Inputs are organized into two banks of four, plus one bank of two. Each bank has an isolated common terminal. In the case of DC input signals, the input may be wired in as either the sourcing or sinking type. 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 eight current sinking DC output channels use terminals on the top connector. The three common terminals are internally connected, meaning all outputs actually share the same electrical common. The wiring example above shows all commons connected together, because it is best to share the common current among the three terminal connections. Note the requirement for external power on the end (right-most) terminal. The equivalent output circuit shows one channel of the bank of eight.

Auxiliary +24V Power Supply

The F1-130AD has a +24V supply output to power external devices. The output is rated at 0.5 Amperes, and includes short-circuit protection and full isolation from internal CPU circuitry. These features make it ideal for powering sensors, solenoids, and other field devices. In fact, it can be used as the supply for loads in the DC output circuits. Since the outputs are the sinking type, you'll need to connect +24V to the output commons. Be sure the combined load currents do not exceed 0.5 A. Note that on the F1-130AD, the +24V auxiliary output is not high enough to power its input circuits (input ON threshold is 90VDC).

F1-130AD General Specifications

External Power Requirements	100 – 240 + 10% -15%
Communication Port	K-Sequence, 9600 baud, 8 data bits, odd parity
Programming cable type	D2-DSCBL
Internal Field Supply Ratings	+24VDC , 0.5A maximum, isolated
Operating Temperature	32 to 140° F (0 to 60° 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 AWG14 or two AWG16, AWG24 minimum

AC Input Specifications

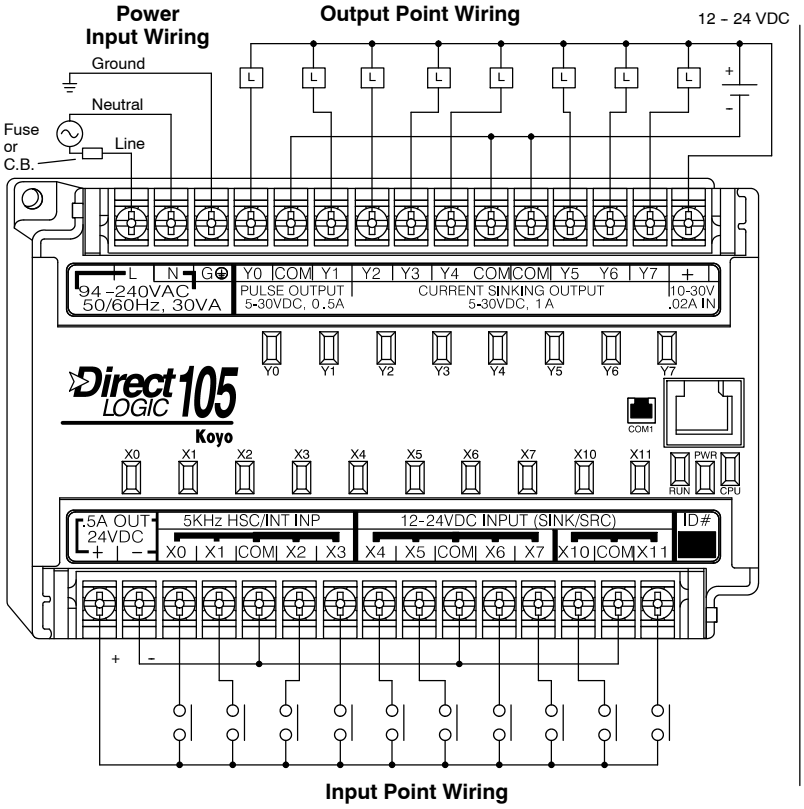
Input Voltage Range for ON condition	80 – 132 VAC, or 90 – 150 VDC
Input Current	6 mA @ 132 VAC 6.8 mA @ 150 VDC
Maximum Voltage	132 VAC, or 150 VDC
ON Current/Voltage	>4 mA @ 80 VAC, or 90 VDC
OFF Current/Voltage	<2 mA @ 45 VAC, or 60 VDC
OFF to ON Response	< 8 mS
ON to OFF Response	<15 mS
Status Indicators	Logic Side
Commons	4 channels / common x 2 banks, 2 channels / common x 1 bank

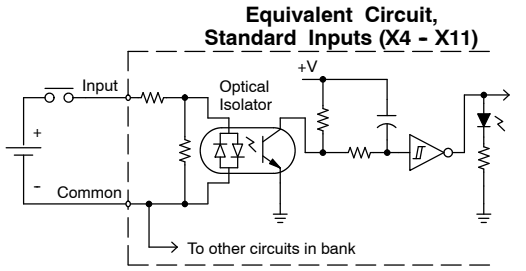
DC Output Specifications

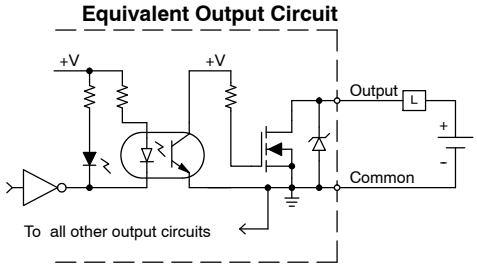
Parameter	Pulse Outputs, Y0 – Y1	Standard Outputs, Y2 – Y7
Operating Voltage	5 – 30 VDC	5 – 30 VDC
Peak Voltage	60 VDC (7 kHz maximum frequency)	60 VDC
On Voltage Drop	0.4 VDC @ 0.25A	0.4 VDC @ 0.5A
Max Current (resistive)	0.5 A / point (subject to derating)	1.0 A / point (subject to derating)
Max leakage current	15 μA @ 30 VDC	15 μA @ 30 VDC
Max inrush current	1.5 A for 10 mS, 0.5 A for 100 mS	3 A for 10 mS, 1 A for 100 mS
Extenal DC power required	10 – 30 VDC @30 mA, plus load current	10 – 30 VDC @30 mA, plus load current
OFF to ON Response	<10 μS	3.5 μS
ON to OFF Response	<70 μS	110 μS
Status Indicators	Logic Side	Logic Side
Commons	Internally connected	Internally connected
Fuses	None	None

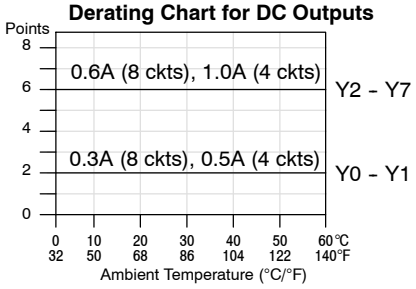
F1-130DD/ F1-130DD-CE I/O Wiring Diagram

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

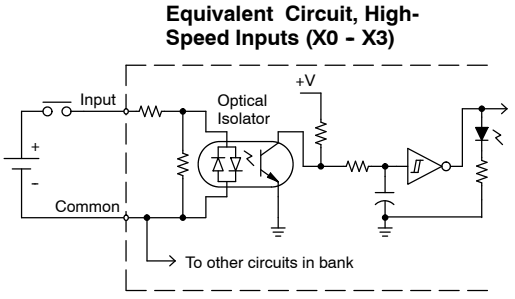








The ten DC input channels use terminals on the bottom connector. Inputs are organized into two banks of four, plus one bank of two. Each bank has an isolated common terminal, and may be wired as either sinking or sourcing inputs. The wiring example above shows all commons connected together, but separate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



The eight current sinking DC output channels use terminals on the top connector. Outputs are organized as one bank of sinking outputs. The three common terminals are internally connected, so all outputs actually share the same electrical common. The wiring example above shows all commons connected together, because it is best to share the common current among the three terminal connections. The equivalent output circuit shows one channel of the bank of eight.

Auxiliary +24V Power Supply

These versions have a +24V supply output to power external devices. The output is rated at 0.5 Amperes, and includes short-circuit protection and full isolation from internal circuitry. These features make it ideal for powering sensors, solenoids, and

other field devices. In fact, it can be used as the DC supply for switches or sensors in the input circuit, or for loads in the DC output circuits (up to 0.5 A).

F1-130DD/ F130-DD-CE General Specifications

External Power Requirements	100 – 240 +10% -15%
Communication Port	K-Sequence, 9600 baud, 8 data bits, odd parity
Programming cable type	D2-DSCBL
Internal Field Supply Ratings	+24VDC , 0.5A maximum, isolated
Operating Temperature	32 to 140° F (0 to 60° 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 AWG14 or two AWG16, AWG24 minimum

DC Input Specifications

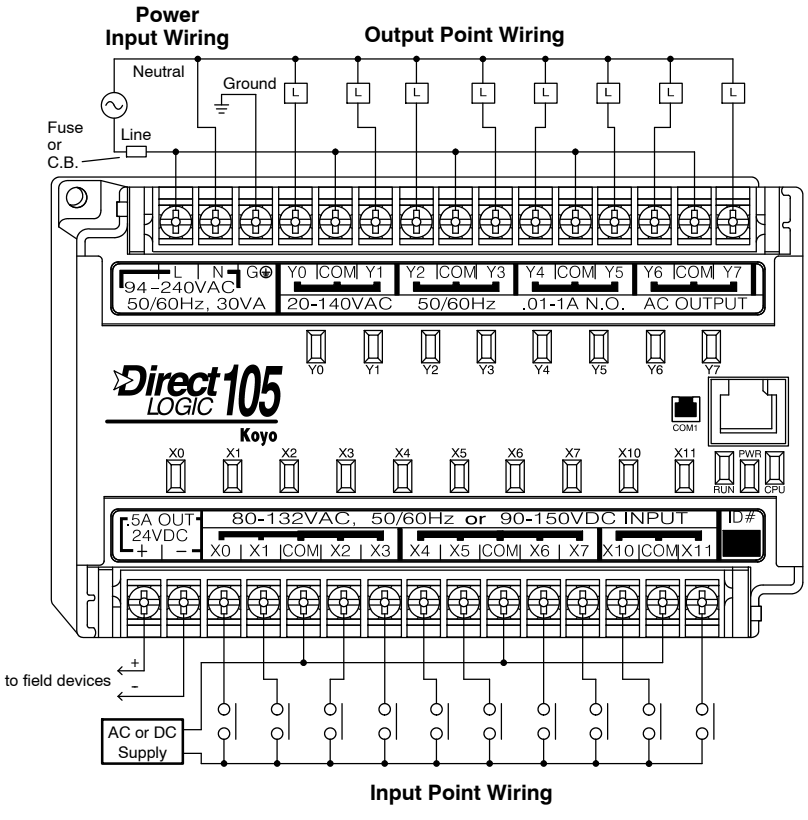
Parameter	High-Speed Inputs, X0 – X3	Standard DC Inputs X4 – X11
Input Voltage Range	10 – 26.4 VDC	10 – 26.4 VDC or 21.6 – 26.4 VAC
Maximum Voltage	30 VDC (5 kHz maximum frequency)	30 VDC
Minimum Pulse Width	100 μs	N/A
ON Voltage Level	> 9.0 VDC	> 9.0 VDC
OFF Voltage Level	< 2.0 VDC	< 2.0 VDC
Input Impedance	2.8 kΩ @ 12 – 24 VDC	2.8 kΩ @ 12 – 24 VDC
Minimum ON Current	>3 mA	>3 mA
Maximum OFF Current	< 0.5 mA	<0.5 mA
OFF to ON Response	<50 μS	2 – 8 mS, 4 mS typical
ON to OFF Response	< 50 μS	2 – 8 mS, 4 mS typical
Status Indicators	Logic side	Logic side
Commons	4 channels / common x 1 bank	4 channels / common x 1 bank, 2 channels / common x 1 bank

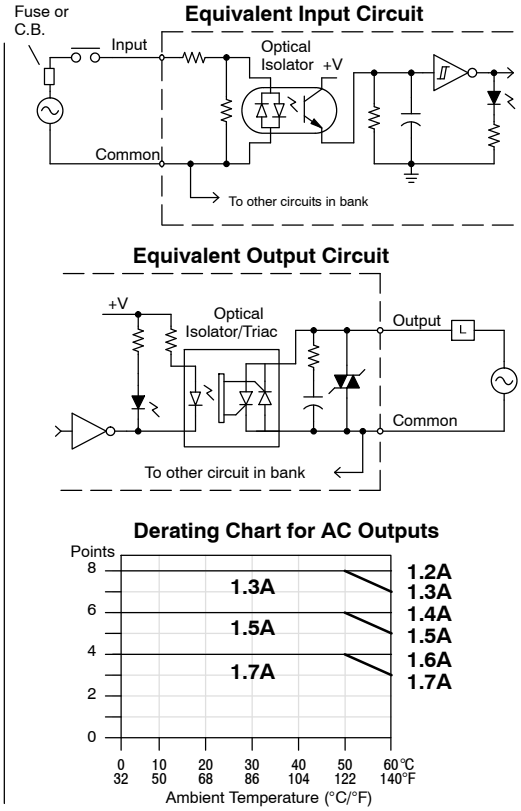
DC Output Specifications

Parameter	Pulse Outputs, Y0 – Y1	Standard Outputs, Y2 – Y7
Operating Voltage	5 – 30 VDC	5 – 30 VDC
Peak Voltage	60 VDC (7 kHz maximum frequency)	60 VDC
On Voltage Drop	0.4 VDC @ 0.25A	0.4 VDC @ 0.5A
Max Current (resistive)	0.5 A / point (subject to derating)	1.0 A / point (subject to derating)
Max leakage current	15 μA @ 30 VDC	15 μA @ 30 VDC
Max inrush current	1.5 A for 10 mS, 0.5 A for 100 mS	3 A for 10 mS, 1 A for 100 mS
External DC power required	10 – 30 VDC @30 mA, plus load current	10 – 30 VDC @30 mA, plus load current
OFF to ON Response	<10 μs	3.5 μs
ON to OFF Response	<70 μs	110 μs
Status Indicators	Logic Side	Logic Side
Commons	Internally connected	Internally connected
Fuses	None	None

F1-130AA I/O Wiring Diagram

The F1-130AA Micro PLC features ten AC inputs and eight AC outputs. The following diagram shows a typical field wiring example. The AC external power connection uses three terminals at the top left as shown.





The ten AC input channels use terminals on the bottom connector. This input type also works for high-voltage DC signals. Inputs are organized into two banks of four, plus one bank of two. Each bank has an isolated common terminal. In the case of DC input signals, the input may be wired in as either the sourcing or sinking type. 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 eight output channels use terminals on the top connector. Outputs are organized into four banks of two 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.

Auxiliary +24V Power Supply

The F1-130AA has a +24V supply output to power external devices. The output is rated at 0.5 Amperes, and includes short-circuit protection and full isolation from internal CPU circuitry. These features make it ideal for powering sensors, solenoids, and other field devices. Note that on the F1-130AA, the +24V auxiliary output cannot directly power its input and output circuits(input ON threshold is 90VDC, outputs require AC only).

F1-130AA General Specifications

External Power Requirements	100 – 240 +10% -15%
Communication Port	K-Sequence, 9600 baud, 8 data bits, odd parity
Programming cable type	D2-DSCBL
Internal Field Supply Ratings	+24VDC , 0.5A maximum, isolated
Operating Temperature	32 to 140° F (0 to 60° 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 AWG14 or two AWG16, AWG24 minimum

AC Input Specifications

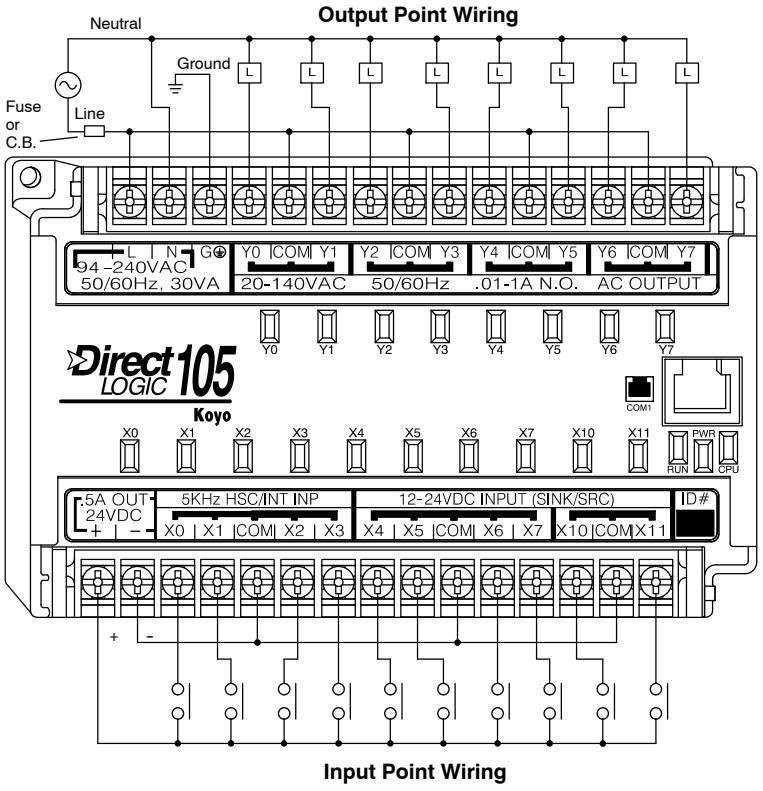
Input Voltage Range for ON condition	80 – 132 VAC, or 90 – 150 VDC
Input Current	6 mA @ 132 VAC 6.8 mA @ 150 VDC
Maximum Voltage	132 VAC, or 150 VDC
ON Current/Voltage	>4 mA @ 80 VAC, or 90 VDC
OFF Current/Voltage	<2 mA @ 45 VAC, or 60 VDC
OFF to ON Response	< 8 mS
ON to OFF Response	<15 mS
Status Indicators	Logic Side
Commons	4 channels / common x 2 banks, 2 channels / common x 1 bank

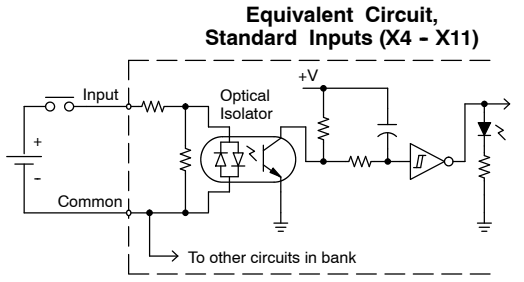
AC Output Specifications

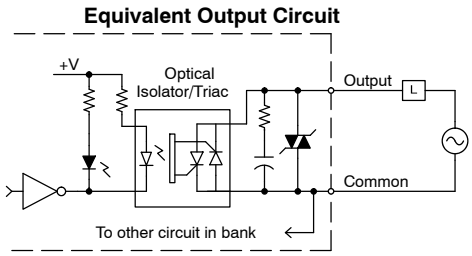
Operating Voltage	20 – 140 VAC, 47 – 63 Hz
Peak Voltage	400 VAC
On Voltage Drop	1.3 VAC @ 2 A
Max Current	1.7 A / point, subject to derating
Max leakage current	1 mA @ 400 VAC
Max inrush current	30 A for 10 mS, 15 A for 100 mS
Minimum Load	10 mA
OFF to ON Response	8.33 mS @ 60 Hz, zero-crossing, 10 mS @ 50 Hz, zero-crossing
ON to OFF Response	8.33 mS @ 60 Hz, zero-crossing, 10 mS @ 50 Hz, zero-crossing
Status Indicators	Logic Side
Commons	2 channels / common x 4 banks
Fuses	None (external recommended)

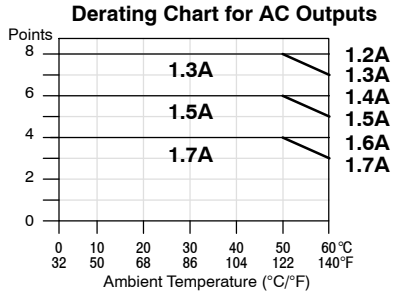
F1-130DA I/O Wiring Diagram

The F1-130DA Micro PLC features ten DC inputs and eight AC outputs. The following diagram shows a typical field wiring example. The AC external power connection uses three terminals at the top left as shown.

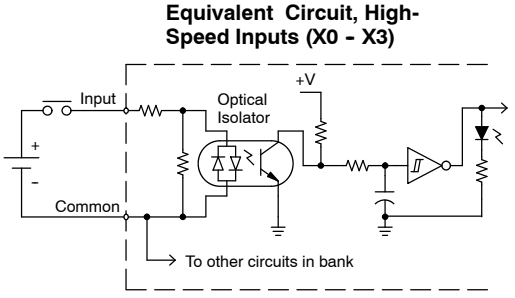








The ten DC input channels use terminals on the bottom connector. Inputs are organized into two banks of four, plus one bank of two. Each bank has an isolated common terminal, and may be wired as sinking or sourcing inputs. The wiring example above shows all commons connected together, but separate supplies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



The eight output channels use terminals on the top connector. Outputs are organized into four banks of two 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.

Auxiliary +24V Power Supply

The F1-130DA has a +24V supply output to power external devices. The output is rated at 0.5 Amperes, and includes short-circuit protection and full isolation from internal CPU circuitry. These features make it ideal for powering sensors, solenoids, and other field devices. In fact, it can be used as the DC supply for switches or

sensors in the input circuit. Note that on the F1-130DA, the +24V output cannot power its output circuits, because they require AC voltages.

F1-130DA General Specifications

External Power Requirements	100 - 240 +10% -15%
Communication Port	K-Sequence, 9600 baud, 8 data bits, odd parity
Programming cable type	D2-DSCBL
Internal Field Supply Ratings	+24VDC , 0.5A maximum, isolated
Operating Temperature	32 to 140° F (0 to 60° 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 AWG14 or two AWG16, AWG24 minimum

DC Input Specifications

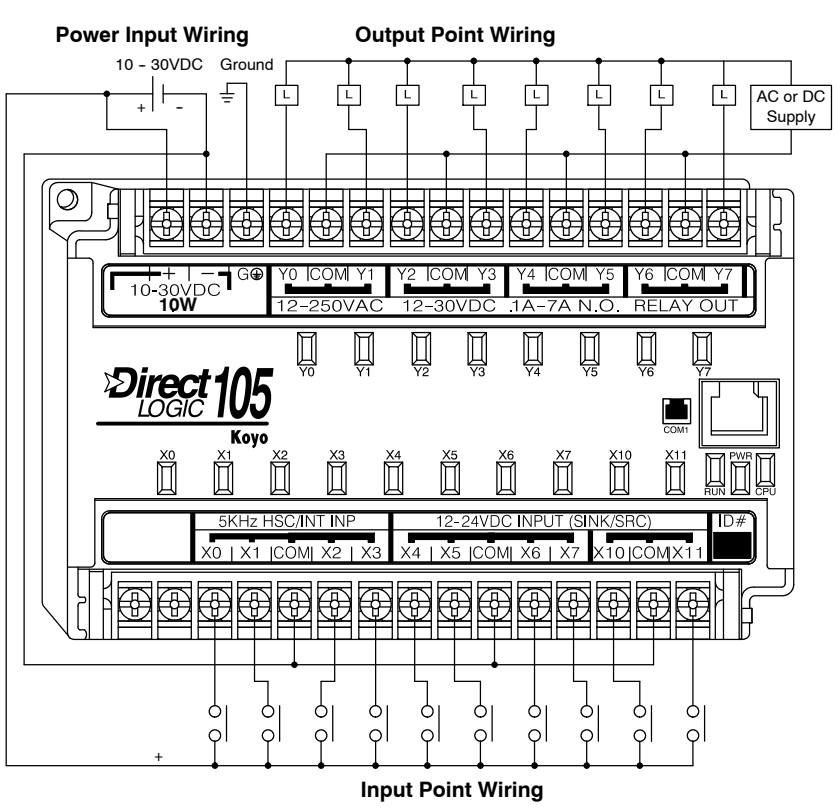
Parameter	High-Speed Inputs, X0 - X3	Standard DC Inputs X4 - X11
Input Voltage Range	10 - 26.4 VDC	10 - 26.4 VDC or 21.6 - 26.4 VAC
Maximum Voltage	30 VDC (5 kHz maximum frequency)	30 VDC
Minimum Pulse Width	100 μS	N/A
ON Voltage Level	> 9.0 VDC	> 9.0 VDC
OFF Voltage Level	< 2.0 VDC	< 2.0 VDC
Input Impedance	2.8 kΩ @ 12 - 24 VDC	2.8 kΩ @ 12 - 24 VDC
Minimum ON Current	>3 mA	>3 mA
Maximum OFF Current	< 0.5 mA	<0.5 mA
OFF to ON Response	<50 μS	2 - 8 mS, 4 mS typical
ON to OFF Response	< 50 μS	2 - 8 mS, 4 mS typical
Status Indicators	Logic side	Logic side
Commons	4 channels / common x 1 bank	4 channels / common x 1 bank, 2 channels / common x 1 bank

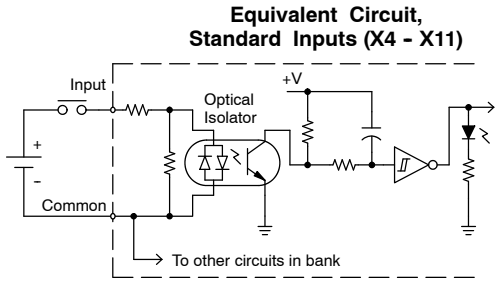
AC Output Specifications

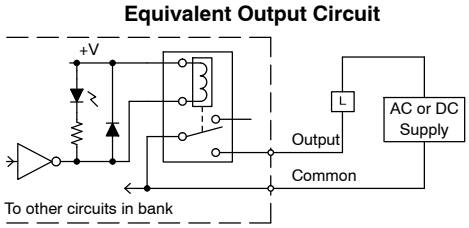
Operating Voltage	20 - 140 VAC, 47 - 63 Hz
Peak Voltage	400 VAC
On Voltage Drop	1.3 VAC @ 2 A
Max Current	1.7 A / point, subject to derating
Max leakage current	1 mA @ 400 VAC
Max inrush current	30 A for 10 mS, 15 A for 100 mS
Minimum Load	10 mA
OFF to ON Response	8.33 mS @ 60 Hz, zero-crossing, 10 mS @ 50 Hz, zero-crossing
ON to OFF Response	8.33 mS @ 60 Hz, zero-crossing, 10 mS @ 50 Hz, zero-crossing
Status Indicators	Logic Side
Commons	2 channels / common x 4 banks
Fuses	None (external recommended)

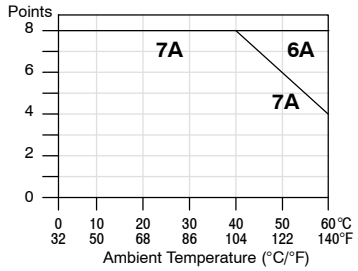
F1-130DR-D I/O Wiring Diagram

The F1-130DR-D Micro PLC features ten DC inputs and eight relay outputs. The following diagram shows a typical field wiring example. The DC external power connection uses three terminals at the top left as shown.

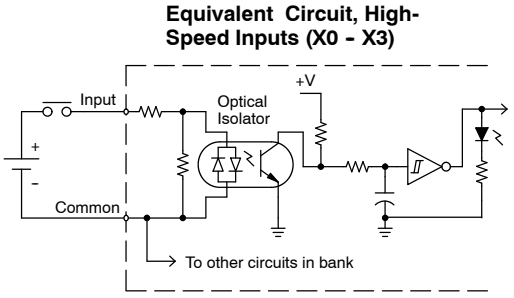








The ten DC input channels use terminals on the bottom connector. Inputs are organized into two banks of four, plus one bank of two. Each bank has an isolated common terminal, and may be wired as sinking or sourcing inputs. The wiring example above shows all commons connected together, but separate supplies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



The eight output channels use terminals on the top connector. Outputs are organized into four banks of two 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.

No Auxiliary +24V Power Supply

The F1-130DR-D does not include a +24V output, as do most other DL105 PLCs. Since this unit requires +24V as the main supply input, it it usually most economical to use the same supply to power suitable field devices. In the wiring diagram above,

the external power source for the unit also powers the input circuitry. The same external supply can power both input and output circuits, because they are both isolated from the internal logic circuitry.

F1-130DR-D General Specifications

External Power Requirements	10-30VDC, 1.5A
Communication Port	K-Sequence, 9600 baud, 8 data bits, odd parity
Programming cable type	D2-DSCBL
Operating Temperature	32 to 140° F (0 to 60° 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 AWG14 or two AWG16, AWG24 minimum

DC Input Specifications

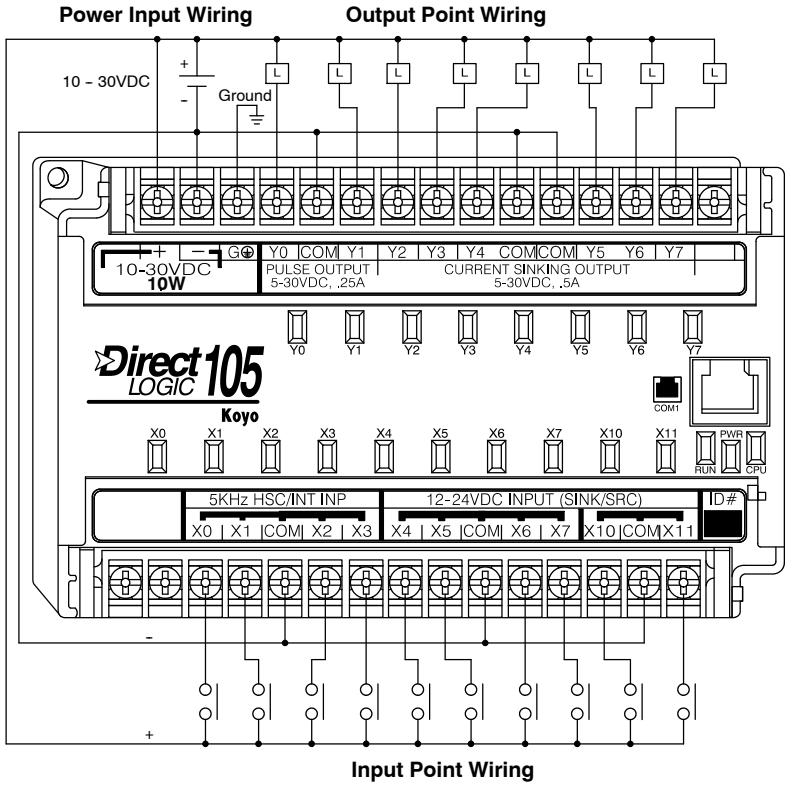
Parameter	High-Speed Inputs, X0 - X3	Standard DC Inputs X4 - X11
Input Voltage Range	10 - 26.4 VDC	10 - 26.4 VDC or 21.6 - 26.4 VAC
Maximum Voltage	30 VDC (5 kHz maximum frequency)	30 VDC
Minimum Pulse Width	100 μs	N/A
ON Voltage Level	> 9.0 VDC	> 9.0 VDC
OFF Voltage Level	< 2.0 VDC	< 2.0 VDC
Input Impedance	2.8 kΩ @ 12 - 24 VDC	2.8 kΩ @ 12 - 24 VDC
Minimum ON Current	>3 mA	>3 mA
Maximum OFF Current	< 0.5 mA	<0.5 mA
OFF to ON Response	<50 μS	2 - 8 mS, 4 mS typical
ON to OFF Response	< 50 μS	2 - 8 mS, 4 mS typical
Status Indicators	Logic side	Logic side
Commons	4 channels / common x 1 bank	4 channels / common x 1 bank, 2 channels / common x 1 bank

Relay Output Specifications

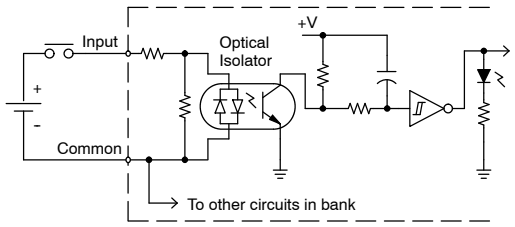
Operating Voltage	12 - 250 VAC, 12 - 30 VDC @ 7A, 30 - 150 VDC @ 0.5A, resistive
Output Current	7A / point (subject to derating) 14A / common
Maximum Motor Load	1/3 HP
Maximum Voltage	265 VAC, 150 VDC
Minimum Off Resistance	100 meg ohms @ 500 VDC
Smallest Recommended Load	10 mA
OFF to ON Response	15 ms
ON to OFF Response	5 ms
Status Indicators	Logic Side
Commons	2 channels / common x 4 banks
Fuses	None (external recommended)

F1-130DD-D I/O Wiring Diagram

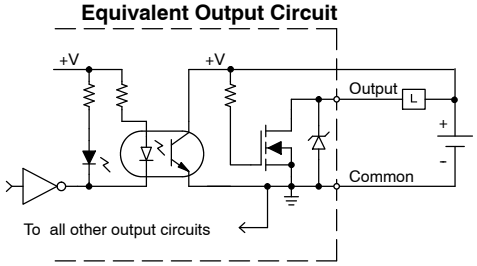
The F1-130DD-D Micro PLC features ten DC inputs and eight DC outputs. The following diagram shows a typical field wiring example. The DC external power connection uses three terminals at the top left as shown.



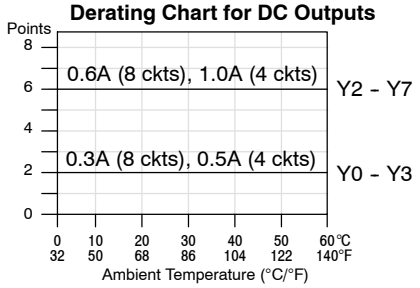
Equivalent Circuit, Standard Inputs (X4 - X11)



Equivalent Output Circuit

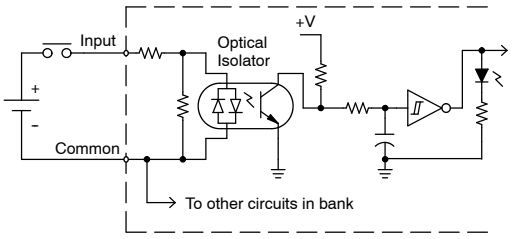


Derating Chart for DC Outputs



The ten DC input channels use terminals on the bottom connector. Inputs are organized into two banks of four, plus one bank of two. Each bank has an isolated common terminal, and can 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 is shown above, and the high-speed input circuit is shown to the right.

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



The eight current-sinking DC output channels use terminals on the top connector. Outputs are organized as one bank of eight. The three common terminals are internally connected, meaning all outputs actually share the same electrical common. The wiring example above shows all commons connected together, because it is best to share the common current among the three terminal connections. The equivalent output circuit shows one channel of the bank of eight.

No Auxiliary +24V Power Supply

The F1-130DR-D does not include a +24V output, as do most other DL105 PLCs. Since this unit requires +24V as the main supply input, it it usually most economical to use the same supply to power suitable field devices. In the wiring diagram above,

the external power source for the unit also powers the input and output circuitry. The same external supply can power both input and output circuits, because they are both isolated from the internal logic circuitry.

F1-130DD-D General Specifications

External Power Requirements	10-30VDC, 1.5A
Communication Port	K-Sequence, 9600 baud, 8 data bits, odd parity
Programming cable type	D2-DSCBL
Operating Temperature	32 to 140° F (0 to 60° 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 AWG14 or two AWG16, AWG24 minimum

DC Input Specifications

Parameter	High-Speed Inputs, X0 - X3	Standard DC Inputs X4 - X11
Input Voltage Range	10 - 26.4 VDC	10 - 26.4 VDC or 21.6 - 26.4 VAC
Maximum 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
Input Impedance	2.8 kΩ @ 12 - 24 VDC	2.8 kΩ @ 12 - 24 VDC
Minimum ON Current	>3 mA	>3 mA
Maximum OFF Current	< 0.5 mA	<0.5 mA
OFF to ON Response	<50 μS	2 - 8 mS, 4 mS typical
ON to OFF Response	< 50 μS	2 - 8 mS, 4 mS typical
Status Indicators	Logic side	Logic side
Commons	4 channels / common x 1 bank	4 channels / common x 1 bank, 2 channels / common x 1 bank

DC Output Specifications

Parameter	Pulse Outputs, Y0 - Y1	Standard Outputs, Y2 - Y7
Operating Voltage	5 - 30 VDC	5 - 30 VDC
Peak Voltage	60 VDC (7 kHz maximum frequency)	60 VDC
On Voltage Drop	0.4 VDC @ 0.25A	0.4 VDC @ 0.5A
Max Current (resistive)	0.5 A / point (subject to derating)	1.0 A / point (subject to derating)
Max leakage current	15 μA @ 30 VDC	15 μA @ 30 VDC
Max inrush current	1.5 A for 10 mS, 0.5 A for 100 mS	3 A for 10 mS, 1 A for 100 mS
Extenal DC power required	10 - 30 VDC @30 mA, plus load current	10 - 30 VDC @30 mA, plus load current
OFF to ON Response	<10 μs	3.5 μs
ON to OFF Response	<70 μs	110 μs
Status Indicators	Logic Side	Logic Side
Commons	Internally connected	Internally connected
Fuses	None	None

Glossary of Specification Terms

Discrete Input	One of ten 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 eight 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.
Terminal Type	Indicates whether the terminal type is a removable or non-removable connector or a fixed terminal.
Status Indicators	The LEDs that indicate the ON/OFF status of an input or output point. All LEDs on DL105 Micro PLCs are electrically located on the logic side of the input or output circuit.

3

High-Speed Input and Pulse Output Features

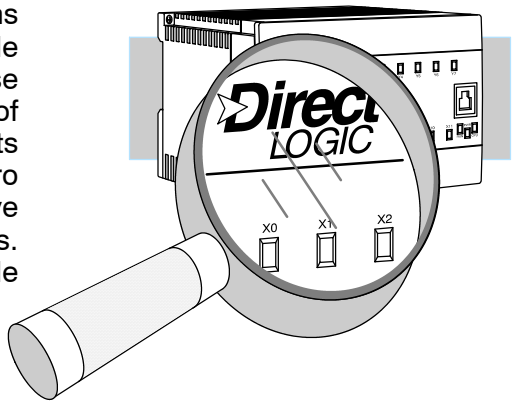
In This Chapter. . . .

— Introduction
— Choosing the HSIO Operating Mode
— Mode 10: High-Speed Counter
— Mode 20: Quadrature Counter
— Mode 30: Pulse Output
— Mode 40: High-Speed Interrupt
— Mode 50: Pulse Catch Input
— Mode 60: Filtered Inputs

Introduction

Built-in Motion Control Solution

Many machine control applications require various types of simple high-speed monitoring and control. These applications usually involve some type of motion control, or high-speed interrupts for time-critical events. The DL105 Micro PLC solves this traditionally expensive problem with built-in CPU enhancements. Let's take a closer look at the available high-speed I/O features.



The available **high-speed input features** are:

- High Speed Counter (5 kHz max.) with up to 24 counter presets and built-in interrupt subroutine, counts up only, with reset
- Quadrature encoder inputs to measure counts and clockwise or counter clockwise direction (5 kHz max.), counts up or down, with reset
- High-speed interrupt input for immediate response to critical or time-sensitive tasks
- Pulse catch feature to monitor one input point, having a pulse width as small as 100μS (0.1ms)
- Programmable discrete filtering (both on and off delay up to 99ms) to ensure input signal integrity (this is the default mode for inputs X0–X3)

The available **pulse output features** are:

- Single-axis programmable pulse output (7 kHz max.) with three profile types, including trapezoidal moves, registration, and velocity control

Availability of HSIO Features

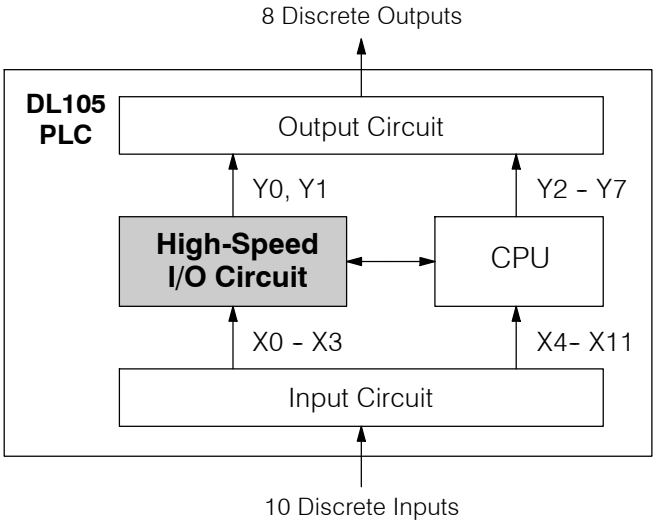
IMPORTANT: Please note the following restrictions on availability of features:

- High-speed input options are available only on DL105s with DC inputs.
- Pulse output options are available only on DL105s with DC outputs.
- Only one HSIO feature may be in use at one time. You cannot use a high–speed input feature and the pulse output at the same time.

DL105 Part Number	Discrete Input Type	Discrete Output Type	High-Speed Input	Pulse Output
F1–130AR	AC	Relay	No	No
F1–130DR	**DC**	Relay	**Yes**	No
F1–130AD	AC	**DC**	No	**Yes**
F1–130DD	**DC**	**DC**	**Yes**	**Yes**
F1–130AA	AC	AC	No	No
F1–130DA	**DC**	AC	**Yes**	No
F1–130DR–D	**DC**	Relay	**Yes**	No
F1–130DD–D	**DC**	**DC**	**Yes**	**Yes**

Dedicated High-Speed I/O Circuit

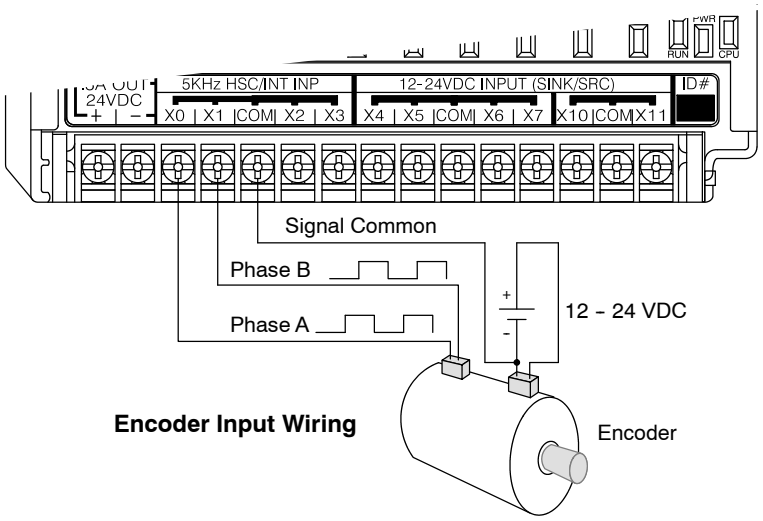
The internal CPU's main task is to execute the ladder program and read/write all I/O points during each scan. In order to service high-speed I/O events, the DL105 includes a special circuit which is dedicated to a portion of the I/O points. Refer to the DL105 block diagram in the figure below.



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

Wiring Diagrams for Each HSIO Mode

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



Encoder Input Wiring

Choosing the HSIO Operating Mode

Understanding the Six Modes

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

Mode Number	Mode Name	Mode Features
10	High-Speed Counter	5 kHz counter with 24 presets and reset input, counts up only, causes interrupt on preset
20	Quadrature Counter	Channel A / Channel B 5 kHz quadrature input, counts up and down
30	Pulse Output	Stepper control – pulse and direction signals, programmable motion profile
40	High-Speed Interrupt	Generates an interrupt based on input transition or time
50	Pulse Catch	Captures narrow pulses on a selected input
60	Discrete/Filtered Input	Rejects narrow pulses on selected inputs

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

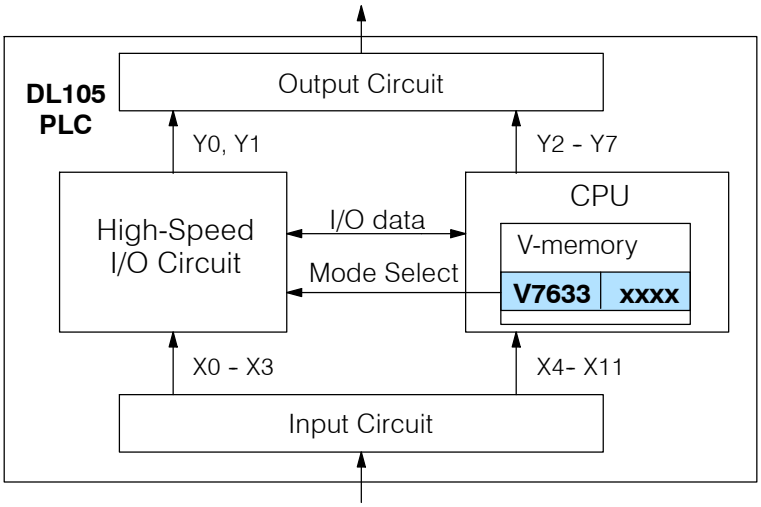
Physical I/O Point Usage						
	DC Input Points				DC Output Points	
Mode	X0	X1	X2	X3	Y0	Y1
High-Speed Counter	Counter clock	Filtered Input	Filtered Input or Reset Cnt	Filtered Input	Regular Output	Regular Output
Quadrature Counter	Phase A Input	Phase B Input	FIltered Input or Reset Cnt	Filtered Input	Regular Output	Regular Output
High-Speed Interrupt	Interrupt Input, or Filtered Input	Filtered Input	Filtered Input	Filtered Input	Regular Output	Regular Output
Pulse Catch	Pulse Input	Filtered Input	Filtered Input	Filtered Input	Regular Output	Regular Output
Pulse Output	Not available	Filtered Input	Filtered Input, or Interrupt to trigger pulse output	Filtered Input	Pulse or CW Pulse	Direction or CCW Pulse
Filtered Input	Filtered Input	Filtered Input	Filtered Input	Filtered Input	Regular Output	Regular Output

Default Mode

Mode 60 (Filtered Inputs) is the default mode. The DL105 is initialized to this mode at the factory, and any time you clear V-memory scratchpad. In the default condition, X0–X3 are filtered inputs (10 mS delay) and Y0–Y1 are standard outputs.

Configuring the HSIO Mode

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



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

Memory Location V7633

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0

2 **0** **5** **0**

Miscellaneous Setup (BCD)

00 = Power Up in Previous Mode
20 = Power Up in Run Mode Always

HSIO Mode Setup (BCD)

00 = Not Used
10 = High-Speed Counting Mode
20 = Quadrature Counting Mode
30 = Pulse Output Train
40 = High-Speed Interrupts
50 = Pulse Catching
60 = Discrete Filtered Inputs (default)

Bits 0 - 7 define the mode number 00, 10.. 60 previously referenced in this chapter. The example data "2050" shown selects Mode 50 - Pulse Catch (BCD = 50) and Power Up in Run Mode (BCD=20). Together they form the 4-digit BCD number 2050.

Configuring Inputs X0 - X3

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

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

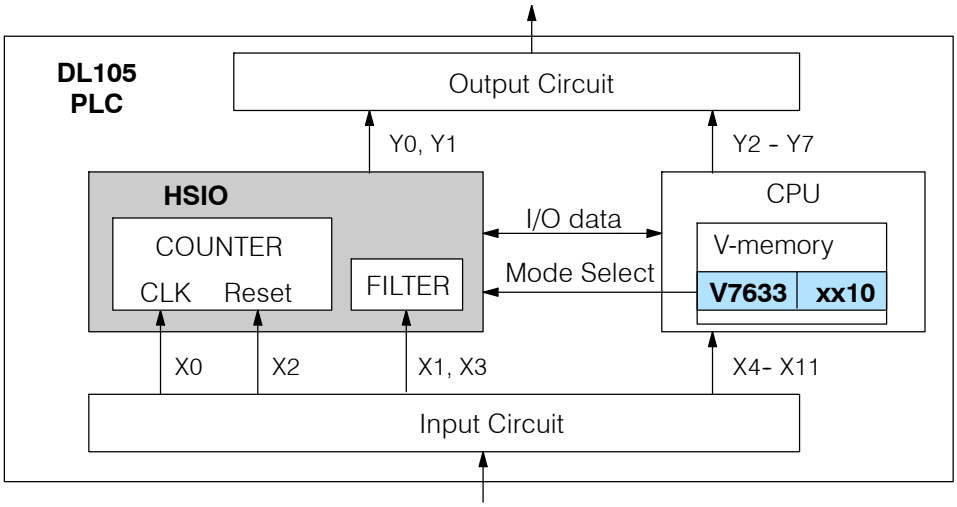
Mode 10: High-Speed Counter

Purpose

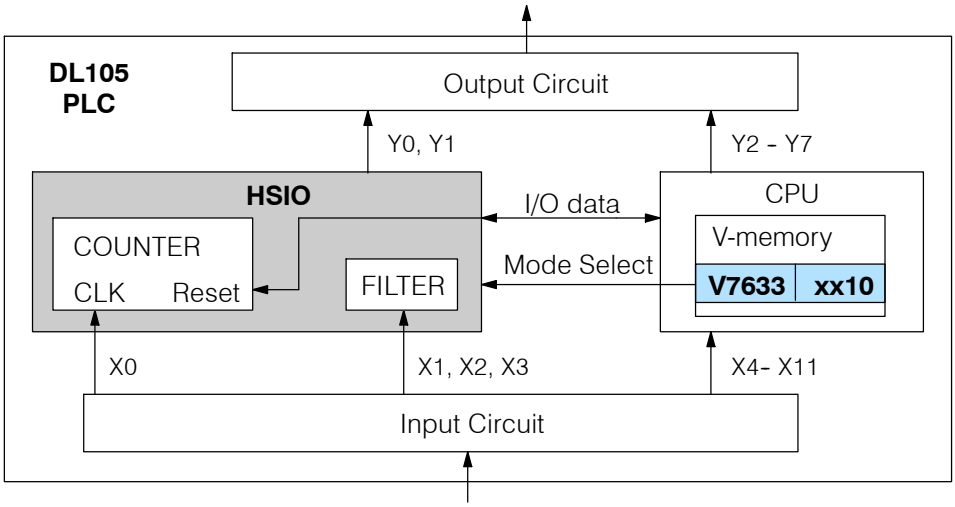
The HSIO circuit contains one high-speed counter. A single pulse train from an external source (X0) clocks the counter on each signal leading edge. The counter counts only upwards, from 0 to 99999999. The counter compares the current count with up to 24 preset values, which you define. The purpose of the presets is to quickly cause an action upon arrival at specific counts, making it ideal for such applications as cut-to-length. It uses counter registers CT76 and CT77 in the CPU.

Functional Block Diagram

Refer to the block diagram below. When the lower byte of HSIO Mode register V7633 contains a BCD "10", the high-speed up counter in the HSIO circuit is enabled. X0 automatically becomes the "clock" input for the high-speed counter, incrementing it upon each off-to-on transition. The external reset input on X2 is the default configuration for Mode 10. Inputs X1 and X3 are filtered inputs, available to the ladder program.



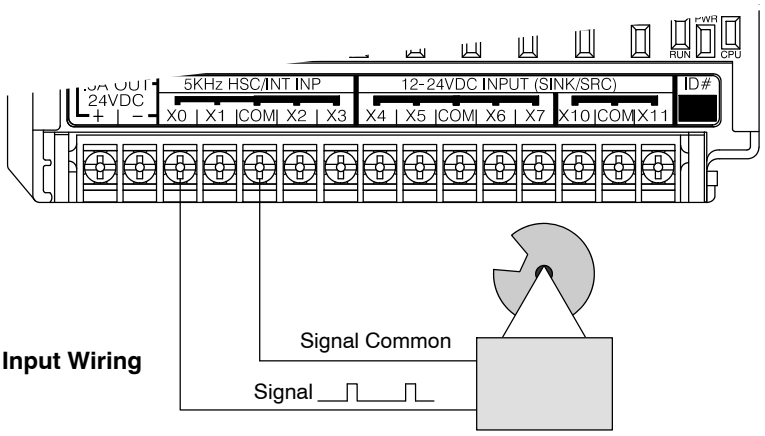
Instead of using X2 as a dedicated reset input, you can configure X2 as a normal filtered input. In this way, the counter reset must be generated in ladder logic.



Next, we will discuss how to program the high-speed counter and its presets.

Wiring Diagram

A general wiring diagram for counters/encoders to the DL105 in HSIO Mode 10 is shown below. Many types of pulse-generating devices may be used, such as proximity switches, single-channel encoders, magnetic or optical sensors, etc. Devices with sinking outputs (NPN open collector) are probably the best choice for interfacing. If the counter sources to the inputs, it must output 12 to 24 VDC. Note that devices with 5V sourcing outputs will not work with DL105 inputs.

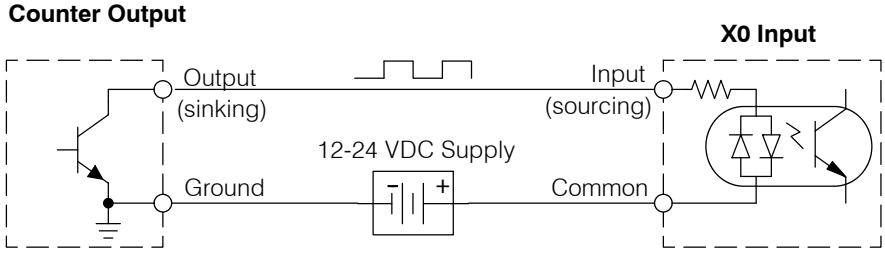


Counter Input Wiring

Interfacing to Counter Outputs

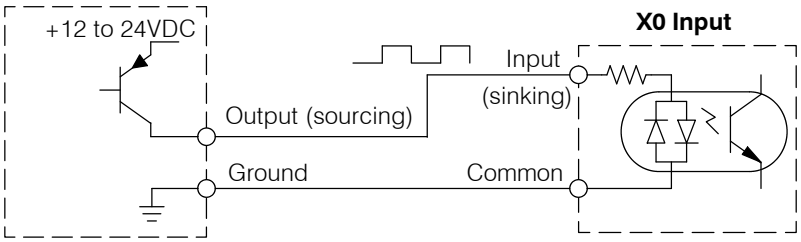
The DL105's DC inputs are flexible in that they detect current flow in either direction, so they can be wired to a counter with either sourcing or sinking outputs. In the following circuit, a counter has open-collector NPN transistor outputs. It sinks current from the PLC input point, which sources current. The power supply can be the +24VDC auxiliary supply or another supply (+12VDC or +24VDC), as long as the input specifications are met.

Counter Output



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

Counter Output



Setup for Mode 10

Recall that V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 10 in the lower byte to select High-Speed Counter Mode. Use BCD 00 or 20 in the upper byte as required. Combine the two bytes into a data word "xx10", for writing to V7633.

Memory Location V7633

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0

2 0 1 0

Miscellaneous Setup (BCD)

00 = Power Up in Previous Mode
20 = Power Up in Run Mode Always

HSIO Mode Setup (BCD)

10 = High-Speed Counter

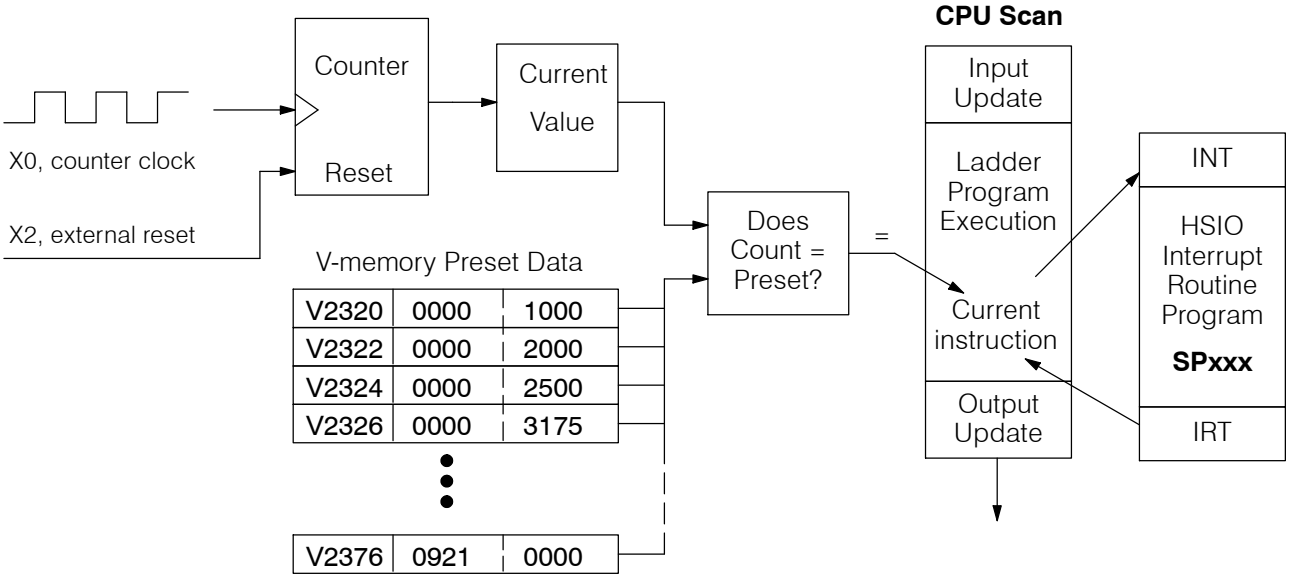
Choose the most convenient method of programming V7633 from the following:

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

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

Presets and Special Relays

The goal of counting is to do a special action when the count reaches a preset value. Refer to the figure below. The counter features 24 presets, which you can program. A preset is a number you derive and store so that the counter will constantly compare the current count with the preset. When the two are equal, a special relay contact is energized and program execution jumps to the interrupt routine. We recommend using the special relay(s) in the interrupt service routine to cause any immediate action you desire. After the interrupt service routine is complete, the CPU returns to the ladder program, resuming program execution from the point of interruption. The compare function is ready for the next preset event.



Preset Data Starting Location

V7630 is a pointer location which points to the beginning of the Preset Data Table. The default starting location for the Preset Data Table is V2320 (default after initializing scratchpad V-memory). However, you may change this by programming a different value in V7630. Use the LDA and OUT instructions as shown:

```
LDA  O2000     Load the octal address, convert to hex, leave result in accumulator.
OUT  V7630     Output this address to V7630, the location of the pointer to the Preset data.
```

Preset Table Pointer

V7630	**2000**

Preset Data

V2000	0000	1000
V2002	0000	2000
V2004	0000	2500
V2006	0000	3175
⋮		
V2076	0000	0000

Using Fewer than 24 Presets

When using fewer than 24 preset registers, the HSIO looks for "0000 FFFF" (use LDD Kffff) in the next preset location to indicate the last preset has been reached. The example to the right uses four presets. The 0000 FFFF in V2331-V2330 indicates the previous preset was the last.

Preset Data

V2320	0000	1000
V2322	0000	2000
V2324	0000	2500
V2326	0000	3175
V2330	0000	FFFF



NOTE: Each successive preset must be greater than the previous preset value. If a preset value is less than a lower-numbered preset value, the CPU cannot compare for that value, since the counter can only count upwards.

Equal Relay Numbers

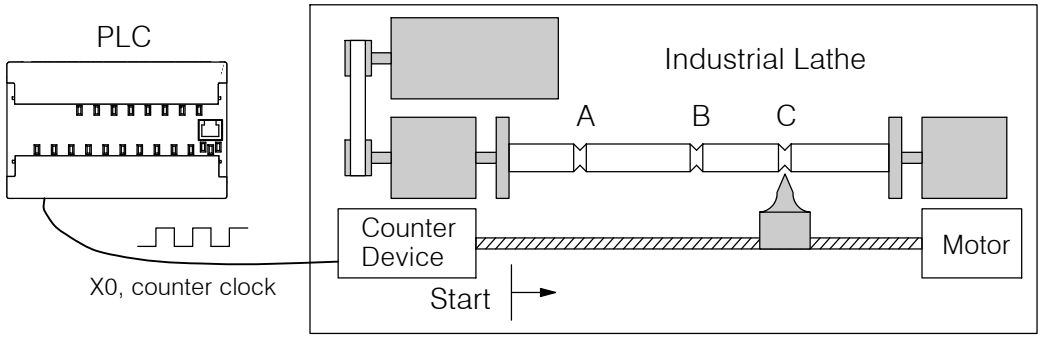
The following table lists all 24 preset register default locations. Each occupies two 16-bit V-memory registers. The corresponding special relay contact number is in the next column. We might also call these "equal" relay contacts, because they are true (closed) when the present high-speed counter value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

Preset	Preset V-memory Register	Special Relay Number	Preset	Preset V-memory Register	Special Relay Number
1	V2321 / V2320	SP540	13	V2351 / V2350	SP554
2	V2323 / V2322	SP541	14	V2353 / V2352	SP555
3	V2325 / V2324	SP542	15	V2355 / V2354	SP556
4	V2327 / V2326	SP543	16	V2357 / V2356	SP557
5	V2331 / V2330	SP544	17	V2361 / V2360	SP560
6	V2333 / V2332	SP545	18	V2363 / V2362	SP561
7	V2335 / V2334	SP546	19	V2365 / V2364	SP562
8	V2337 / V2336	SP547	20	V2367 / V2366	SP563
9	V2341 / V2340	SP550	21	V2371 / V2370	SP564
10	V2343 / V2342	SP551	22	V2373 / V2372	SP565
11	V2345 / V2344	SP552	23	V2375 / V2374	SP566
12	V2347 / V2346	SP553	24	V2377 / V2376	SP567

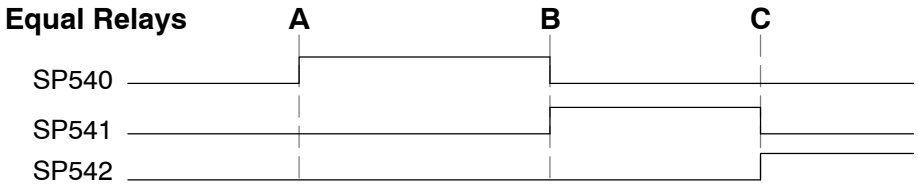
Calculating Your Preset Values

The preset values occupy two data words each. They can range in value from 0000 0000 to 9999 9999, just like the high-speed counter value. All twenty-four values are *absolute* values, meaning that each one is an offset from the counter zero value.

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



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





NOTE: Each successive preset must have a greater value than the previous preset value. In the industrial lathe example, B > A and C > B.

X Input Configuration

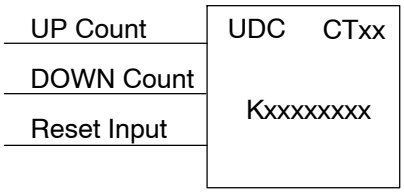
The configurable discrete input options for High-Speed Counter Mode are listed in the table below. Input X0 is dedicated for the counter clock input. Inputs X1 and X3 can only be filtered inputs. The section on Mode 60 operation at the end of this chapter describes programming the filter time constants.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

Input	Configuration Register	Function	Hex Code Required
X0	V7634	Counter Clock	0001
X1	V7635	Filtered Input	xx06 (xx = filter time)
X2	V7636	Counter Reset (no interrupt)	0007
		Counter Reset (with interrupt)	0107
		FIltered Input	xx06 (xx = filter time)
X3	V7637	FIltered Input	xx06 (xx = filter time)

Writing Your Control Program

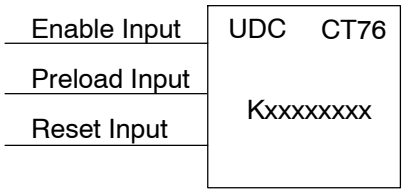
You may recall that the counter instruction is a standard instruction in the DL105 instruction set. Refer to the figure below. The mnemonic for the counter is **UDC** (up-down counter).The DL105 can have up to 64 counters, labeled CT0 through CT77. The high speed counter in the HSIO circuit is accessed in ladder logic by using UDC CT76. It uses counter registers CT76 and CT77 exclusively when the HSIO mode 10 is active (otherwise, CT76 and CT77 are available for standard counter use). The HSIO counter needs two registers because it is a double-word counter. It has three inputs as shown. The first input (Enable) allows counting when active. The middle input (Preload) allows you to change the current count. The bottom signal is the reset. The preload input must be off while the counter is counting.

Standard Counter Function



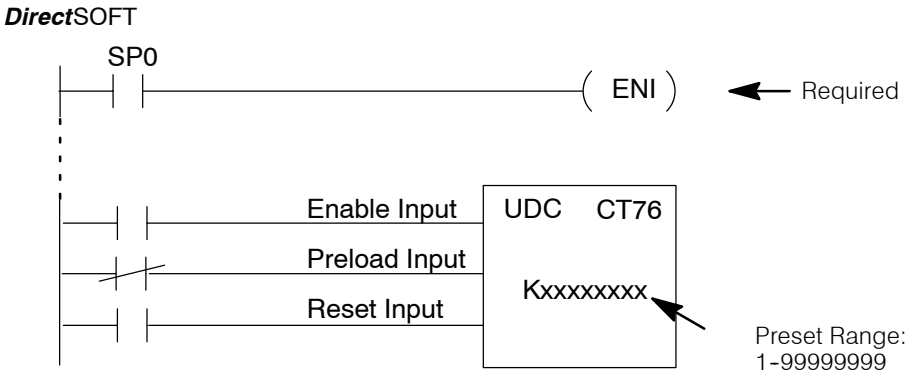
- Counts UP and DOWN
- Preload counter by write to value
- Reset input is internal only

HSIO Counter Function



- Counts UP only
- Can use Preload input to change count
- Reset may be internal or external

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

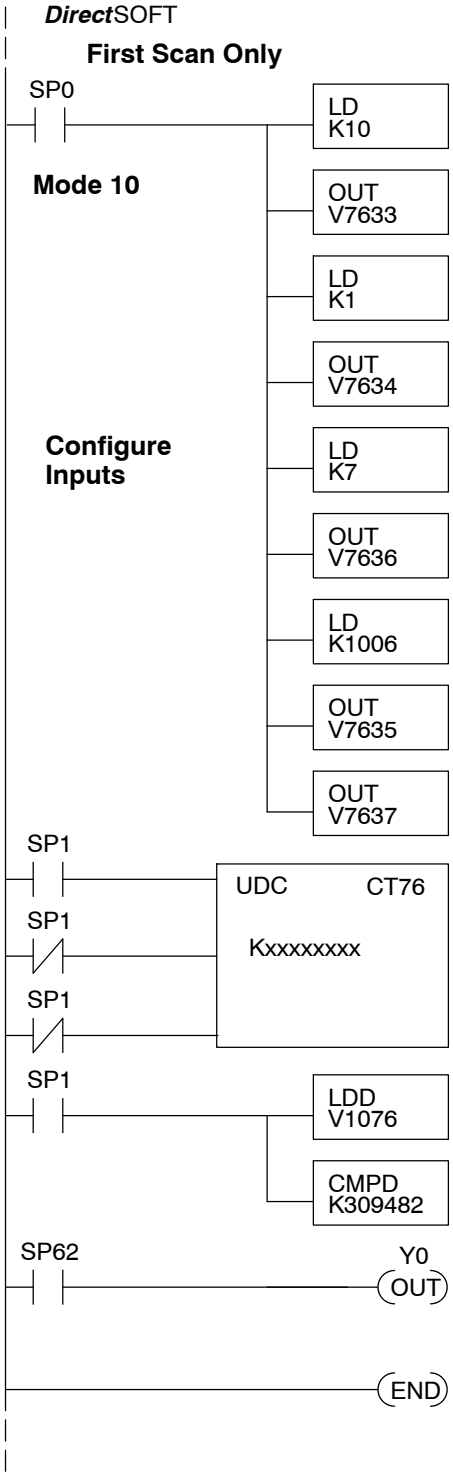


When the enable input is energized, the high-speed counter will respond to pulses on X0 and increment the counter at CT76 – CT77. The reset input contact behaves in a logical OR fashion with the physical reset input X2 (when selected). So,the high speed counter can receive a reset form either the contact(s) on the reset rung in the ladder, OR the external reset X2 if you have configured X2 as an external reset.

Program Example: Counter Without Preset

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

Program Example Cont'd



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

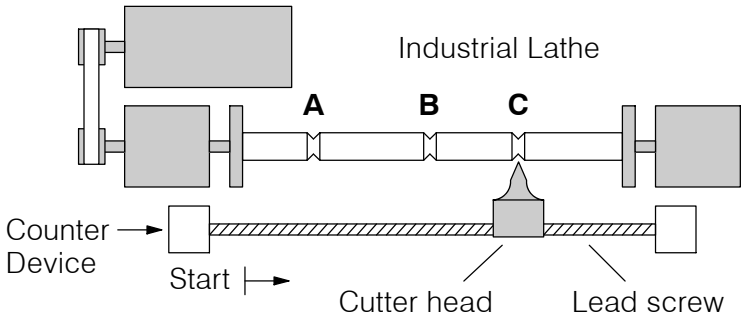
Counter With Presets Program Example

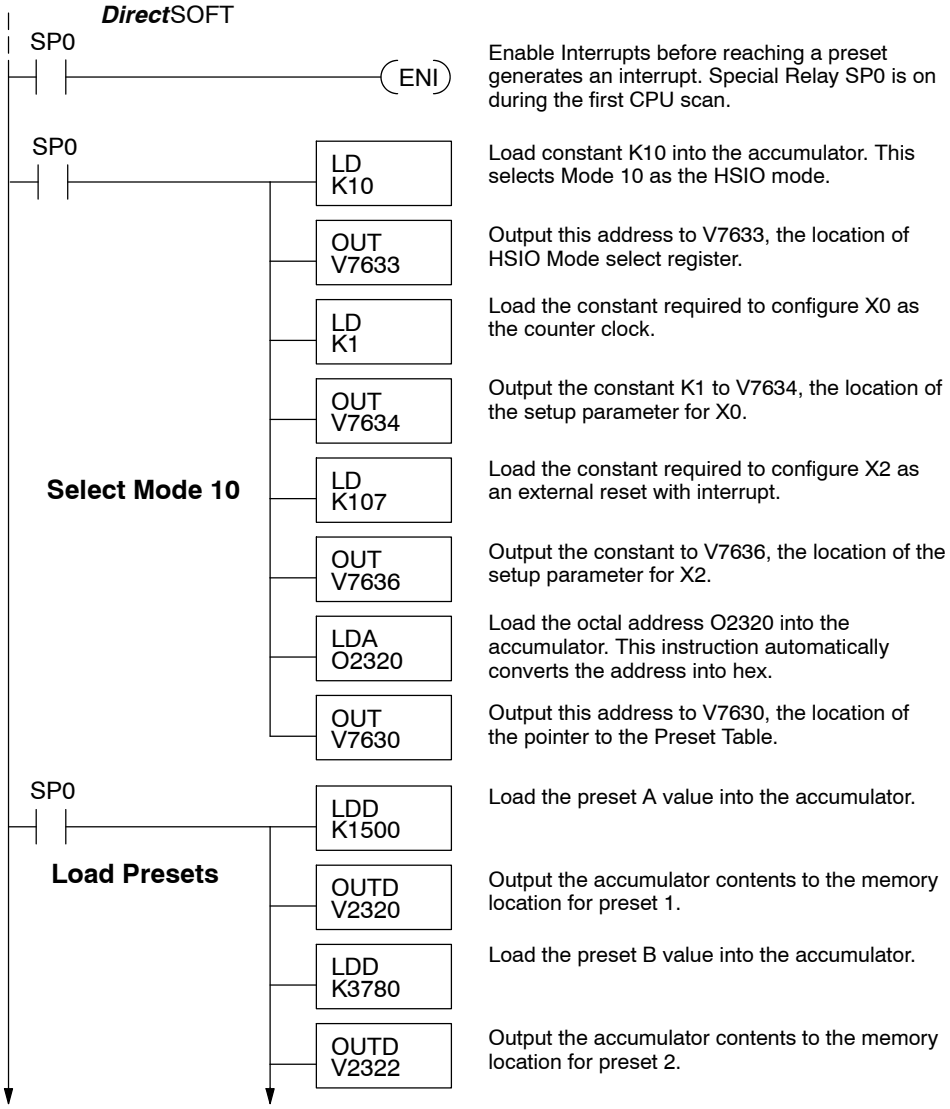
The following example shows how to program the HSIO circuit to trigger on three preset values. You may recall the industrial lathe example from the beginning of this chapter. This example program shows how to control the lathe cutter head to make three grooves in the work-piece at precise positions. When the lead screw turns, the counter device generates pulses which the DL105 can count. The three preset variables A, B, and C represent the positions (number of pulses) corresponding to each of the three grooves.

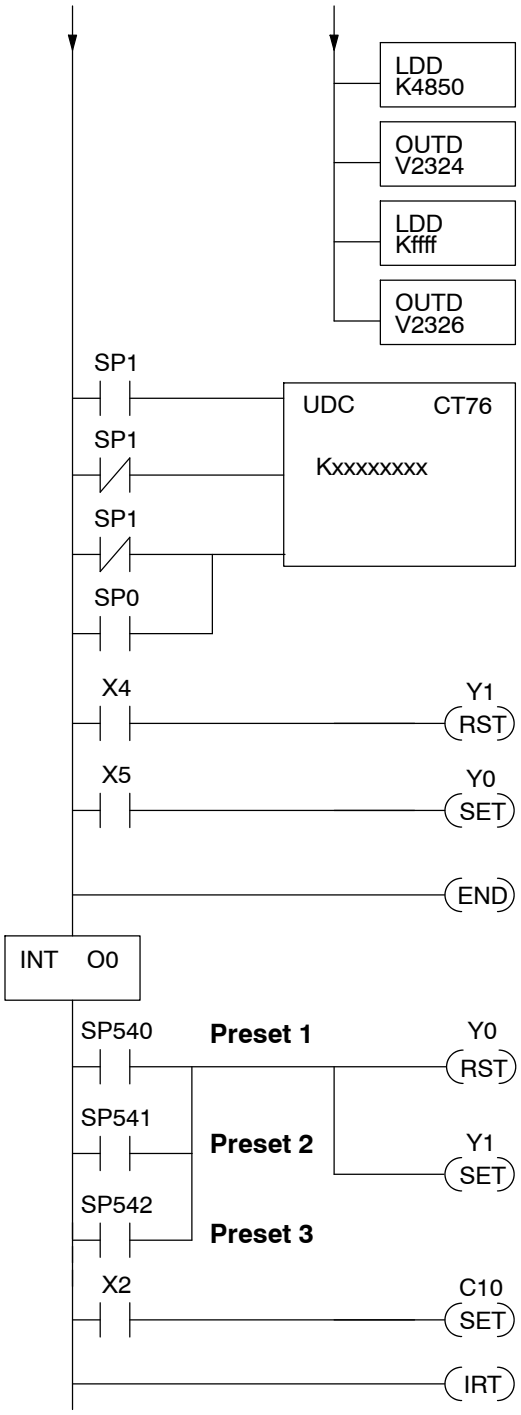
Preset Data

A	V2320	0000	1500
B	V2322	0000	3780
C	V2324	0000	4850
	V2326	0000	FFFF

I/O Assignments
X4 - Cutter head extended
X5 - Cutter head retracted
Y0 - Lead screw motor
Y1 - Cutter head solenoid







Load the preset C value into the accumulator.

Output the accumulator contents to the memory location for preset 3.

Load the constant Kffff into the accumulator. This value represents the end of the preset list.

Output the accumulator contents to the memory location for preset 4 (end of preset marker).

CT76 is the HSIO counter. The first rung's SP1 always enables the counter. The preload contact in the middle is off (unused in this example).

The third rung's Reset input is normally off, because we will use the external reset. You can optionally reset the counter value on each powerup using the SP0 contact.

Input X4 energizes when the groove has finished cutting. So, we retract the cutter head.

Turn lead screw on again, after cutter head has retracted.

END coil marks the end of the main program.

The INT label marks the beginning of the interrupt service routine program.

Inside the interrupt service routine, we turn OFF the lead screw motor immediately.

These special "equal" relays turn on individually as the corresponding preset is reached. In this application, each results in the cutting of a groove (Y1), so they are logically ORed together.

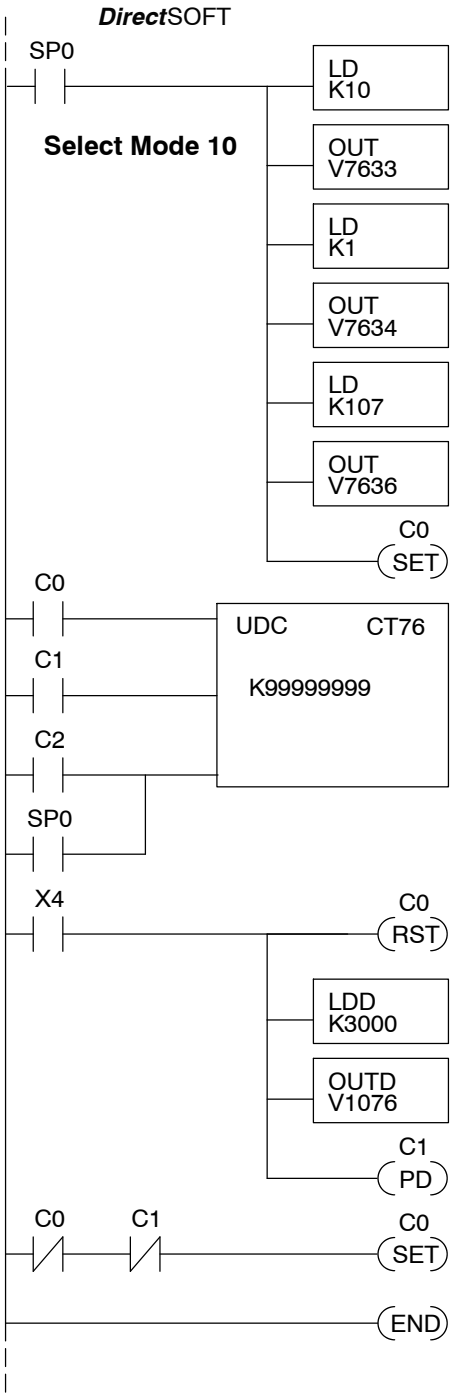
Input X2 will be energized inside the interrupt routine if X2 external interrupt was the source.

Return from the interrupt service routine.

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

Counter With Preload Program Example

The following example shows how you can preload the current count with another value. When the preload command input (X4 in this example) is energized, we disable the counter from counting with C0. Then we write the value K3000 to the count register (V1076-V1077). By pulsing C1 on, we preload the current count of the counter with K3000. When the preload command (X4) is turned off, the counter resumes counting any pulses, but now starting from K3000.



Load constant K10 into the accumulator. This selects Mode 10 as the HSIO mode.

Output this address to V7633, the location of HSIO Mode select register.

Load the constant required to configure X0 as the counter clock.

Output the constant K1 to V7634, the location of the setup parameter for X0.

Load the constant required to configure X2 as an external reset with interrupt.

Output the constant to V7636, the location of the setup parameter for X2.

Set C0 on at powerup to enable counting.

CT76 is the HSIO counter. The first rung's C0 contact enables the counter. The preload contact in the middle is off until the logic near the end of this program turns it on.

The third rung's Reset input is normally off, because we will use the external reset. You can optionally reset the counter value on each powerup using the SP0 contact.

When the preload request is made, the user turns on X4. First we disable counting by resetting C0, the counter's enable input.

Load the BCD value K3000 into the accumulator.

Output the constant to V1076/V1077, the location of the accumulated count for CT76.

Generate a preload counter input pulse, which causes the counter to preload from V1076-V1077.

Enable the counter by setting C0, when the preolad pulse on C1 has occurred (C1 is off).

END coil marks the end of the main program.

Troubleshooting Guide for Mode 10

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

Symptom: The counter does not count.

Possible causes:

1. **Field sensor and wiring** – Verify that the encoder, proximity switch,or counter actually turns on and illuminates the status LED for X0. The problem could be due to sinking-sourcing wiring problem, etc. Remember to check the signal ground connection. Also verify that the pulse on-time is long enough for the PLC to recognize it.
2. **Configuration** – make sure all of the configuration parameters are correct. V7633 must be set to 10, and V7634 must be set to 1 to enable the HSIO counter mode.
3. **Stuck in reset** – check the input status of the reset input, X2. If X2 is on, the counter will not count because it is being held in reset.
4. **Ladder program** – make sure you are using counter CT76 in your program. The top input is the enable signal for the counter. It must be on before the counter will count. The middle input is the preload input and must be off for the counter to count. The bottom input is the counter reset, and must be off during counting.

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

Possible causes:

1. **Configuration** – Ensure the preset values are correct. The presets are 32-bit BCD values having a range of 0 to 99999999. Make sure you write all 32 bits to the reserved locations by using the LDD and OUTD instructions. Use only even–numbered addresses, from V2320 to V2376. If using less than 24 presets, be sure to place "0000FFFF" in the location after the last preset used.
2. **Interrupt routine** – Only use Interrupt #0. Make sure the interrupt has been enabled by executing an ENI instruction prior to needing the interrupt. The interrupt routine must be placed after the main program, using the INT label and ending with an interrupt return IRT.
3. **Special relays** – Check the special relay numbers in your program. Use SP540 for Preset 1, SP541 for Preset 2, etc. Remember that only one special equal relay contact is on at a time. When the counter value reaches the next preset, the SP contact which is on now goes off and the next one turns on.

Symptom: The counter counts up but will not reset.

Possible causes:

1. Check the LED status indicator for X2 to make sure it is active when you want a reset. Or, if you are using an internal reset, use the status mode of ***Direct***SOFT to monitor the reset input to the counter.

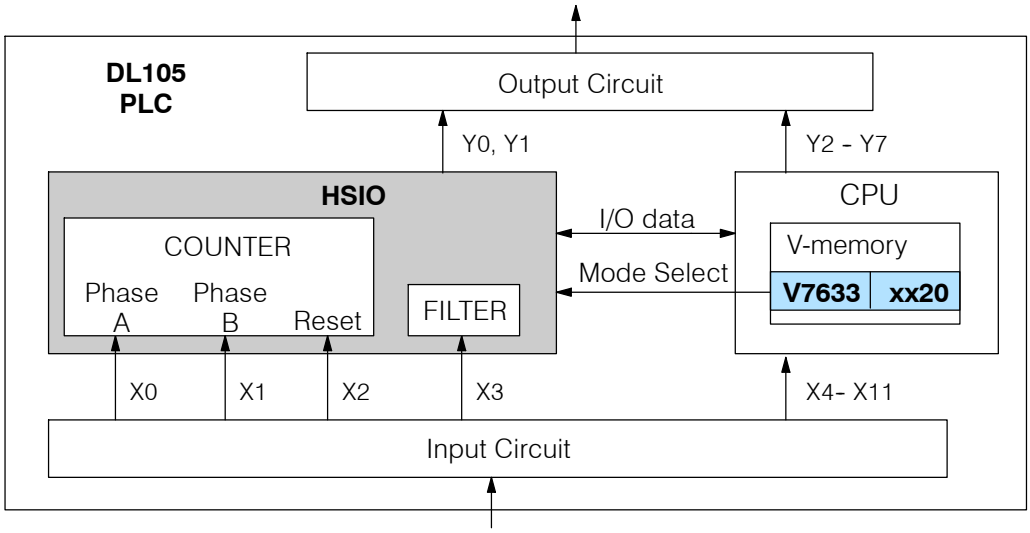
Mode 20: Quadrature Counter

Purpose

The counter in the HSIO circuit can count two quadrature signal pulses instead of a single pulse train (mode 10 operation). Quadrature signals are commonly generated from incremental encoders, which may be rotary or linear. The quadrature counter has a range from 0 to 99999999, and can count at up to a 5 kHz rate, using CT76 and CT77. Unlike Mode 10 operation, the quadrature counter can count UP or DOWN, but *does not feature automated preset values or "interrupt on external reset" capability.* However, you have the standard ladder instruction preset of CT76.

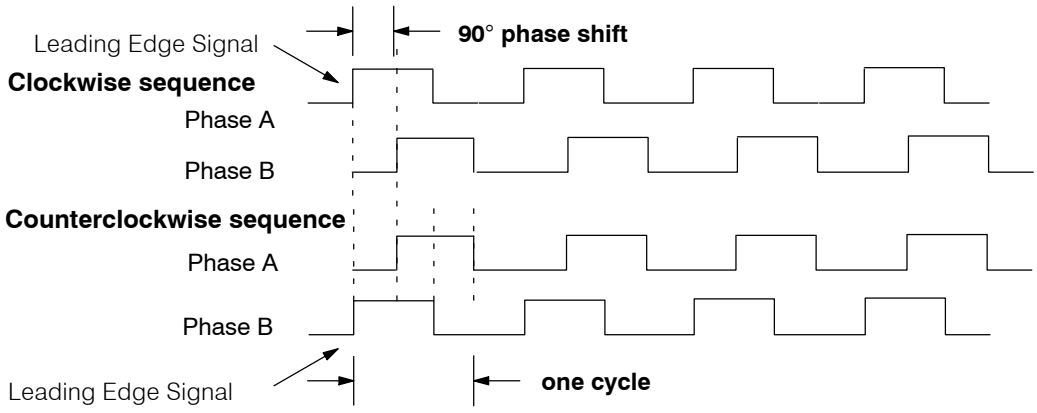
Functional Block Diagram

The diagram below shows HSIO functionality in Mode 20. When the lower byte of HSIO Mode register V7633 contains a BCD "20", the quadrature counter in the HSIO circuit is enabled. Input X0 is dedicated to the Phase A quadrature signal, and input X1 receives Phase B signal. X2 is dedicated to reset the counter to zero value when energized. X3 can only be a regular filtered input in Mode 20.



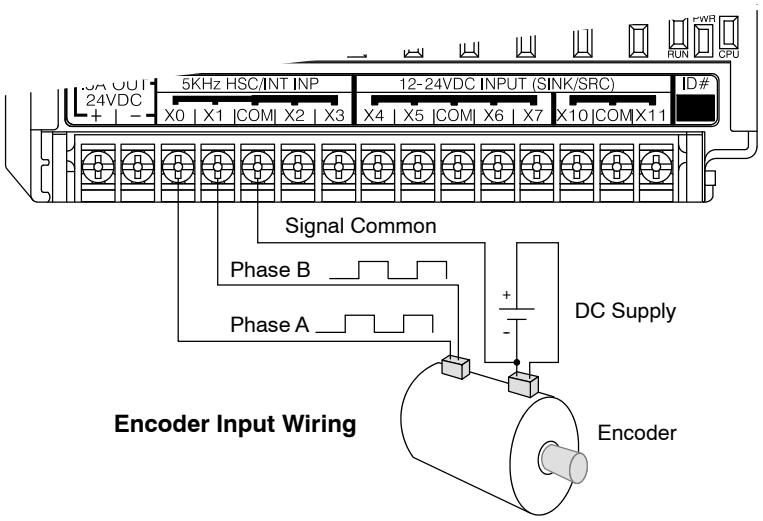
Quadrature Encoder Signals

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



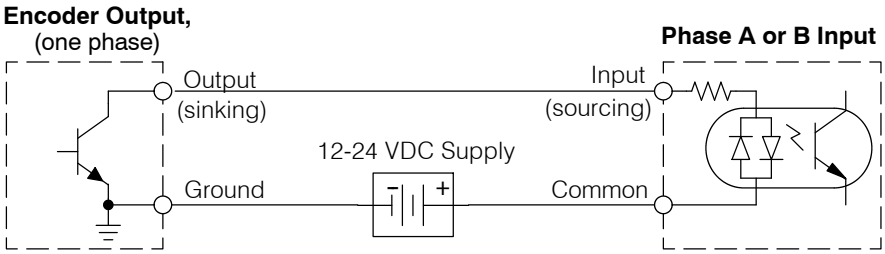
Wiring Diagram

A general wiring diagram for encoders to the DL105 in HSIO Mode 20 is shown below. Encoders with sinking outputs (NPN open collector) are probably the best choice for interfacing. If the encoder sources to the inputs, it must output 12 to 24 VDC. Note that encoders with 5V sourcing outputs will not work with DL105 inputs.

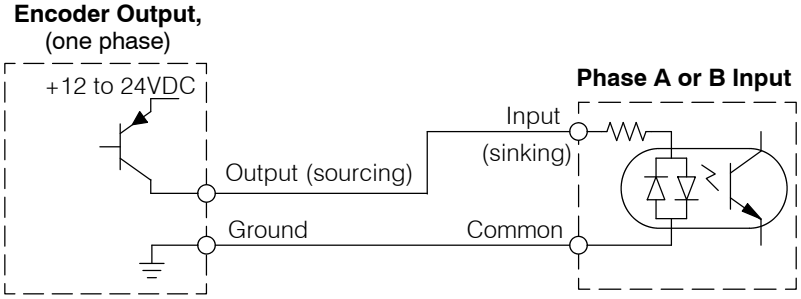


Interfacing to Encoder Outputs

The DL105's DC inputs are flexible in that they detect current flow in either direction, so they can be wired to an encoder with either sourcing or sinking outputs. In the following circuit, an encoder has open-collector NPN transistor outputs. It sinks current from the PLC input point, which sources current. The power supply can be the +24VDC auxiliary supply or another supply (+12VDC or +24VDC), as long as the input specifications are met.

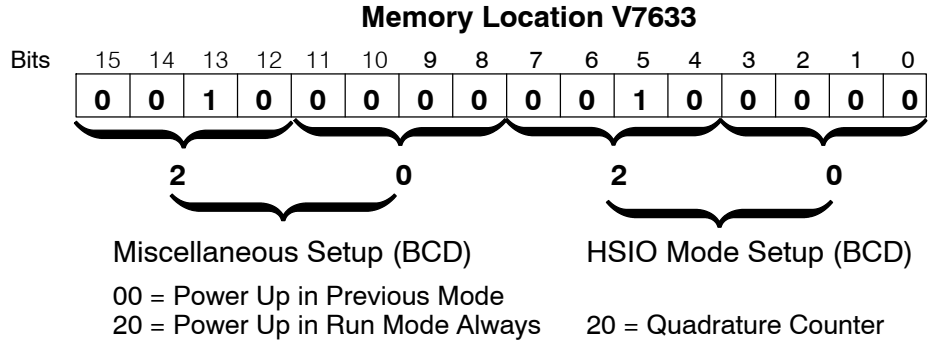


In the next circuit, an encoder has open-emitter PNP transistor outputs. It sources current to the PLC input point, which sinks the current back to ground. Since the encoder sources current, no additional power supply is required. However, note that the encoder output must be 12 to 24 volts (5V encoder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



Setup for Mode 20

Recall that V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 20 in the lower byte to select High-Speed Counter Mode. Use BCD 00 or 20 in the upper byte as required. Combine the two bytes into a data word "xx20", for writing to V7633.



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

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

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

X Input Configuration

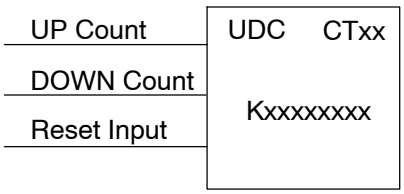
The configurable discrete input options for High-Speed Counter Mode are listed in the table below. Input X0 is dedicated for Phase A, and input X1 is for Phase B. Input X2 is the reset input to the quadrature counter, but it does not cause an interrupt. Input X3 can only be a filtered input. The section on Mode 60 operation at the end of this chapter describes programming the filter time constants.

Input	Configuration Register	Function	Hex Code Required
X0	V7634	Phase A	0012
X1	V7635	Phase B	0000
X2	V7636	Counter Reset (no interrupt)	0007
X3	V7637	Filtered Input	xx06 (xx = filter time)

Writing Your Control Program

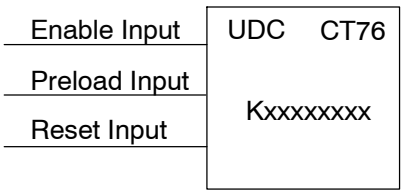
You may recall that the Up-Down counter instruction is standard in the DL105 instruction set. Refer to the figure below. The mnemonic for the counter is **UDC** (up-down counter).The DL105 can have up to 64 counters, labeled CT0 through CT77. The quadrature counter in the HSIO circuit is accessed in ladder logic by using UDC CT76. It uses counter registers CT76 and CT77 exclusively when the HSIO mode 20 is active (otherwise, CT76 and CT77 are available for standard counter use). The HSIO counter needs two registers because it is a double-word counter. It also has three inputs as shown, but they are redefined. The first input is the enable signal, the middle is a preload (write), and the bottom is the reset. The enable input must be on before the counter will count. The preload input allows you to write a new value to the current count. The enable input must be off during a preload.

Standard Counter Function



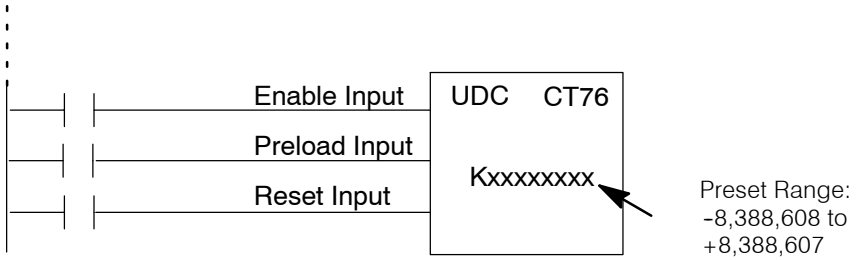
- Counts UP and DOWN
- Preload counter by write to value
- Reset input is internal only

HSIO Counter Function



- Counts UP and DOWN (from X0, X1)
- Can use Preload input to change count
- Reset may be internal or external

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

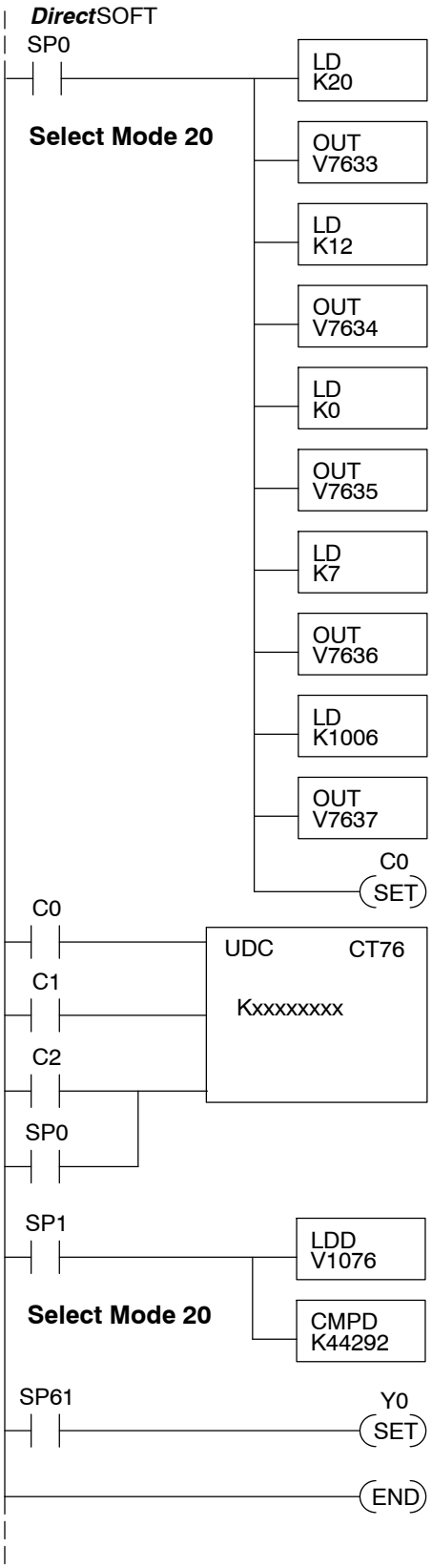


When the enable input is energized, the counter will respond to quadrature pulses on X0 and X1, incrementing or decrementing the counter at CT76 – CT77. The reset input contact behaves in a logical OR fashion with the physical reset input X2. This means the quadrature counter can receive a reset from either the contact(s) on the reset rung in the ladder, OR the external reset X2.

Quadrature Counter w/Preload Program Example

Since presets are not available in quadrature counting, this mode is best suited for simple counting and measuring. The example program on the following page shows how to configure the quadrature counter. The program configures the HSIO circuit for Mode 20 operation, so X0 is Phase A and X1 is Phase B clock inputs.

Program Example Cont'd



Load constant K20 into the accumulator. This selects Mode 20 as the HSIO mode.

Output this address to V7633, the location of the HSIO Mode select register.

Load the constant required to configure X0 as Phase A input.

Output the constant to V7634, the location of the setup register for X0.

Load the constant required to configure X1 as Phase B input.

Output the constant to V7635, the location of the setup register for X1.

Load the constant required to configure X2 as an external reset.

Output the constant to V7636, the location of the setup register for X2.

Load the constant required to configure X3 as a filtered discrete input with a 10 mS time constant.

Output the constant to V7637, the location of the setup register for X3.

Set C0 on at powerup to enable counting.

CT76 is the HSIO quadrature counter. The first rung's SP1 always enables the counter. The preload contact in the middle is controlled by the logic below this counter instruction.

The third rung's Reset input is normally off, because we will use the external reset. You can optionally reset the counter value on each powerup using the SP0 contact.

Load the current value of the counter into the accumulator on each scan.

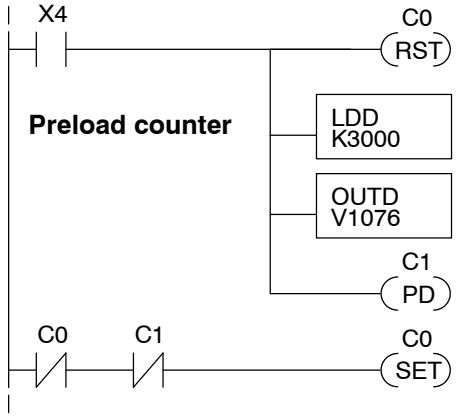
Compare the value in the accumulator with the constant K44292. If they are equal, the SP61 contact will be turned on.

Set Y1 to ON when the counter reaches our comparison value.

END coil marks the end of the main program..

To preload the counter, just add the following example rungs to the program above.

Counter Preload Program Example



When the preload request is made, the user turns on X4. First we disable counting by resetting C0, the counter's enable input.

Load the BCD value K3000 into the accumulator.

Output the constant to V1076/V1077, the location of the accumulated pulse count.

Generate a preload counter input pulse, which causes the counter to preload from V1076-V1077.

Enable the counter by setting C0, when the preolad pulse on C1 has occurred (C1 is off).

Troubleshooting Guide for Mode 20

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

Symptom: The counter does not count.

Possible causes:

1. **Field sensor and wiring** – Verify that the encoder or other field device inputs actually turn on and illuminates the status LEDs for X0 and X1. A standard incremental encoder will visibly, alternately turn on the LEDs for X0 and X1 when rotating slowly (1 RPM). Or, the problem could be due to a sinking-sourcing wiring problem, etc. Remember to check the signal ground connection. Also verify that the pulse on-time, duty cycle, voltage level, and frequency are within the input specifications.
2. **Configuration** – make sure all of the configuration parameters are correct. V7633 must be set to 20, and V7634 must be set to "0012" to enable the Phase A input, and V7635 must be set to "0000" to enable the Phase B input.
3. **Stuck in reset** – check the input status of the reset input, X2. If X2 is on, the counter will not count because it is being held in reset.
4. **Ladder program** – make sure you are using counter CT76 in your program. The top input is the enable signal for the counter. It must be on before the counter will count. The middle input is the preload input and must be off for the counter to count. The bottom input is the counter reset, and must be off during counting.

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

Possible causes:

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

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

Possible causes:

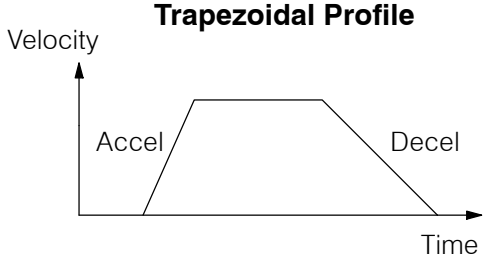
1. Check the LED status indicator for X2 to make sure it is active when you want a reset. Also verify the configuration register V7636 for X2 is set to 7. Or, if you are using an internal reset, use the status mode of ***Direct***SOFT to monitor the reset input to the counter.

Mode 30: Pulse Output

Purpose

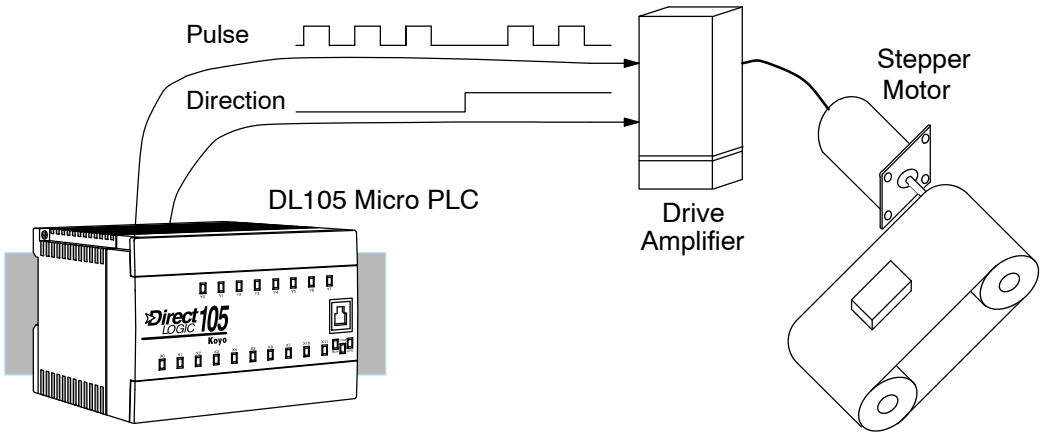
The HSIO circuit in Mode 30 generates output pulse trains suitable for open-loop control of a single-axis motion positioning system. It generates pulse (stepper increment) and direction signals which you can connect to motor drive systems and perform various types of motion control. Using Mode 30 Pulse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

Trapezoidal Profile

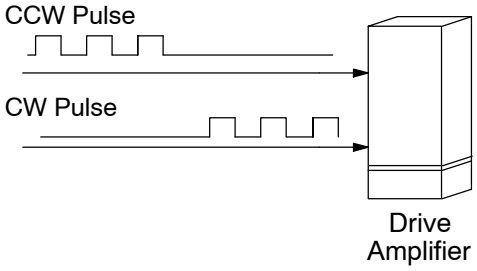


- **Trapezoidal** – Accel Slope to Target Velocity to Decel Slope
- **Registration** – Velocity to Position Control on Interrupt (also used for home search moves)
- **Velocity Control** – Speed and Direction only

The HSIO circuit becomes a high-speed pulse generator (up to 7 kHz) in Mode 30. By programming acceleration and deceleration values, position and velocity target values, the HSIO function automatically calculates the entire motion profile. The figure below shows the DL105 generating pulse and direction signals to the drive amplifier of a stepper positioning system. The pulses accomplish the profile independently and without interruption to ladder program execution in the CPU.



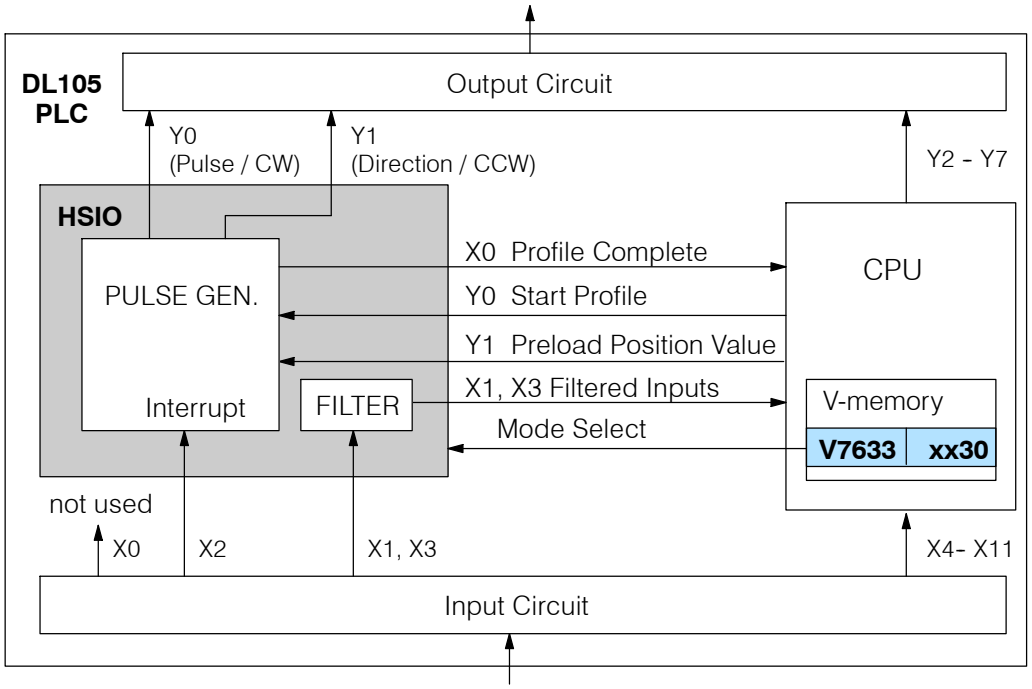
In the figure above, the DL105 generates pulse and direction signals. Each pulse represents the smallest increment of motion to the positioning system (such as one step or micro-step to a stepper system). Alternatively, the HSIO Pulse Output Mode may be configured to deliver counter clock-wise (CCW) and clock-wise (CW) pulse signals as shown to the right.



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

Functional Block Diagram

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



IMPORTANT NOTE: In Pulse Output Mode, X0, Y0, and Y1 references are redefined or are used differently in two ways. *Physical* references refer to terminal screws, while *logical* references refer to I/O references in the ladder program. Please read the items below to understand this very crucial point.

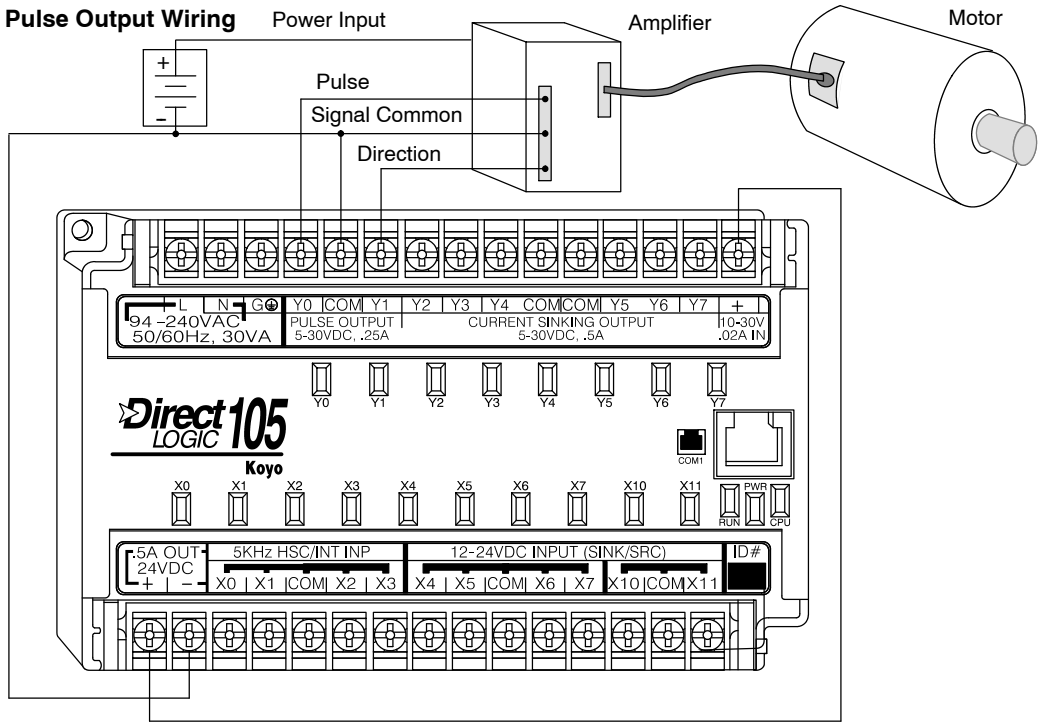
Notice the I/O point assignment and usage in the above diagram:

- Physical input X0 is not used, so the terminal screw will not be wired. However, the HSIO function uses logical reference X0 for "Profile Complete" contact, which is available for ladder program use.
- X1 and X3 can only be filtered inputs in Pulse Output Mode, and they are available as input contacts to the ladder program.
- X2 behaves as an external interrupt to the pulse generator for registration profiles. In other profile modes, it can be used as a filtered input just like X1 and X3 (registration mode configuration shown above).
- References "Y0" and "Y1" are used in two different ways. At the output connector, Y0 and Y1 terminals deliver the pulses to the motion system. The ladder program uses logical references Y0 and Y1 to initiate "Start Profile" and "Load Position Value" HSIO functions in Mode 30.

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

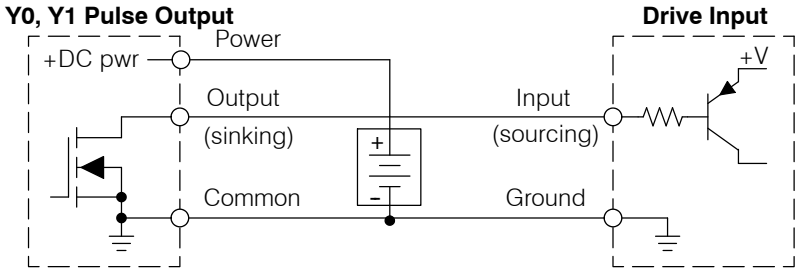
Wiring Diagram

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

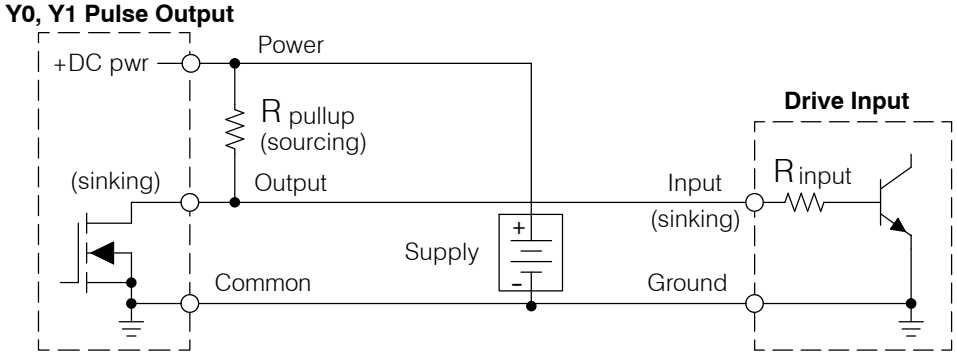


Interfacing to Drive Inputs

The pulse signals from Y0 and Y1 outputs will typically go to drive input circuits as shown above. Remember that the DL105's DC outputs are sinking-only. It will be helpful to locate equivalent circuit schematics of the drive amplifier. The following diagram shows how to interface to a sourcing drive input circuit.



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



Motion Profile Specifications

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

Parameter	Specification
Profiles	Trapezoidal – Accel Slope / Target Velocity / Decel Slope
	Registration – Velocity to Position Control on Interrupt
	Velocity Control – Speed and Direction only
Position Range	–88388608 to 88388607
Positioning	Absolute / relative command
Velocity Range	40 Hz to 7 kHz
V-memory registers	V2320 to V2325 (Profile Parameter Table)
Current Position	CT76 and CT77 (V1076 and V1077)

Physical I/O Configuration

The configurable discrete I/O options for Pulse Output Mode are listed in the table below. Physical input X0 is not available, because the CPU uses logical X0 contact to sense "profile complete". Therefore, V7634 is put to another use: selecting pulse/direction or CCW/CW modes for the pulse outputs. Inputs X1 and X3 can only be filtered inputs. Input X2 is dedicated as the external interrupt for use in registration mode.

Physical Input	Configuration Register	Function	Hex Code Required
–	V7634	Y0 = Pulse Y1 = Direction	0103
		Y0 = CW Pulse Y1 = CCW Pulse	0003
X1	V7635	Filtered Input	xx06 (xx = filter time)
X2	V7636	External Interrupt	1006
X3	V7637	Filtered Input	xx06 (xx = filter time)

Logical I/O Functions

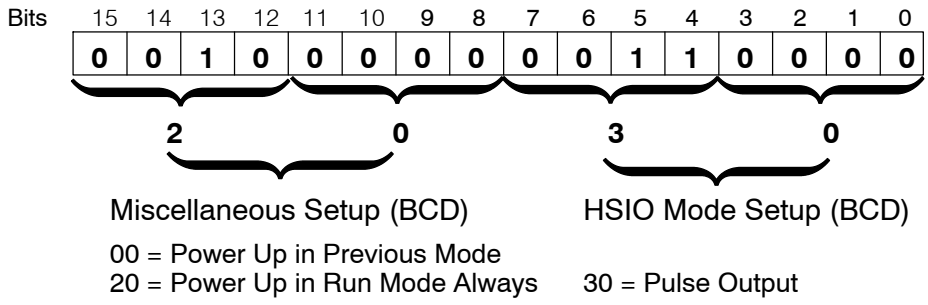
The following logical I/O references define functions that allow the HSIO to communicate with the ladder program.

Logical I/O	Function
X0	Profile Complete – the HSIO turns on X0 to the CPU when the profile completes. Goes back off when Start Profile (Y0) turns on.
Y0	Start Profile – the ladder program turns on Y0 to start motion. If turned off before the move completes, motion stops. Turning it on again will start another profile, unless the current position equals the target position.
Y1	Preload Position Value – if motion is stopped and Start Profile is off, you can load a new value in CT76/CT77, and turn on Y1. At that transition, the value in CT76/CT77 becomes the current position.

Setup for Mode 30

Recall that V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 30 in the lower byte to select Pulse Output Mode. Use BCD 00 or 20 in the upper byte as required. Combine the two bytes into a data word "xx30", for writing to V7633.

Memory Location V7633



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

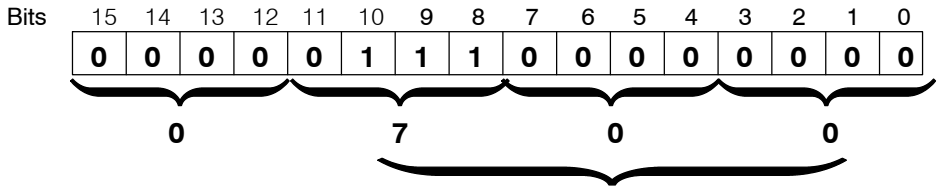
- Include load and out instructions in your ladder program
- ***Direct***SOFT's memory editor
- Use the Handheld Programmer D2-HPP

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

Profile / Velocity Select Register

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

Memory Location V2320 (default)



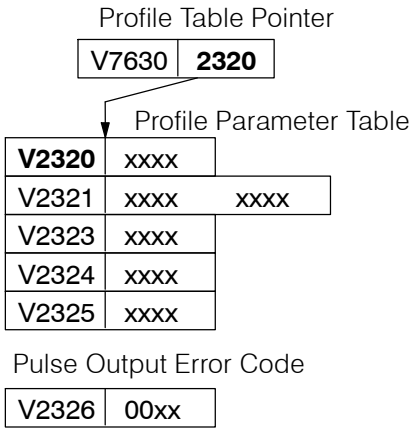
The ladder program must program this location before initiating any of the three profiles. The LD and OUT instruction will write all 16 bits, so be sure to fully specify the full four-digit BCD value for the Position / Velocity Select Register each time.

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

Profile Parameter Table

V7630 is a pointer location which points to the beginning of the Profile Parameter Table. The default starting location for the profile parameter table is V2320. However, you may change this by programming a different value in V7630. Remember to use the LDA (load address) instruction, converting octal into hex.

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



Trapezoidal Profile

V-Memory	Function	Range	Units
V2320, bits 12–15	Trapezoidal Profile	0=absolute, 8=relative	–
V2320, bits 0–11	Target Velocity Value	4 to 700	x 10 pps
V2321/ 2322	Target Position Value	-88388608 to 88388607	Pulses
V2323	Starting Velocity	4 to 100	x 10 pps
V2324	Acceleration Time	1 to 100	x 100 mS
V2325	Deceleration Time	1 to 100	x 100 mS
V2326	Error Code	(see end of section)	–

Registration Profile

V-Memory	Function	Range	Units
V2320, bits 12–15	Registration Profile	9=relative	–
V2320, bits 0–11	Target Velocity Value	4 to 700	x 10 pps
V2321/ 2322	Target Position Value	-88388608 to 88388607	Pulses
V2323	Starting Velocity	4 to 100	x 10 pps
V2324	Acceleration Time	1 to 100	x 100 mS
V2325	Deceleration Time	1 to 100	x 100 mS
V2326	Error Code	(see end of section)	–

Velocity Profile

V-Memory	Function	Range	Units
V2320	Velocity Profile	2000 only	–
V2321/ 2322	Direction Select	80000000=CCW, 0=CW	Pulses
V2323	Velocity	4 to 700	x 10 pps
V2326	Error Code	(see end of section)	–

Choosing the Profile Type

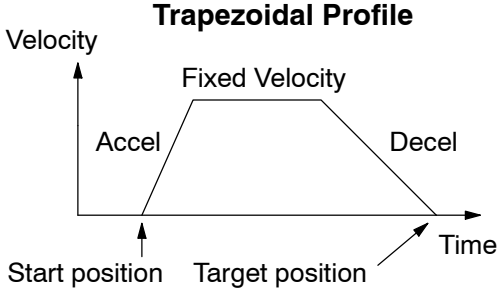
Pulse Output Mode generates three types of motion profiles. Most applications use one type for most moves. However, each move can be different if required.

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

Trapezoidal Profile Defined

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

Trapezoidal Profile



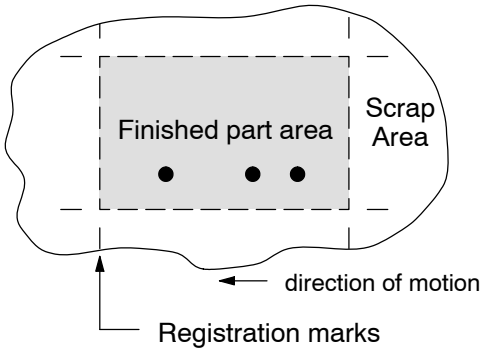
Trapezoidal profiles are best for simple point-to-point moves, when the distance between the starting and ending positions of the move is known in advance.

Registration and Home Search Profiles Defined

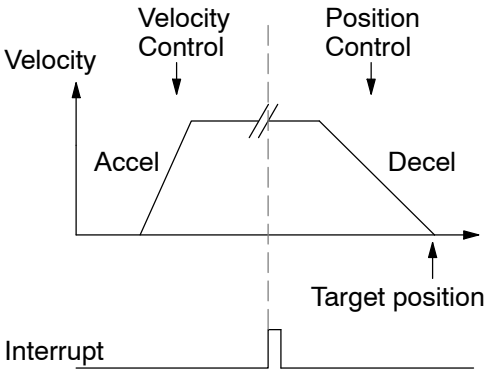
Registration profiles solve a class of motion control problems. In some applications, product material in work moves past a work tool such as a drill station. Shown to the right, registration marks on the scrap area of the work-piece allow a machine tool to register its position relative to the rectangle,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

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

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



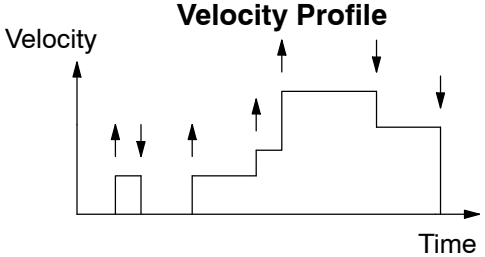
Registration Profile



Velocity Profile Defined

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

Velocity Profile

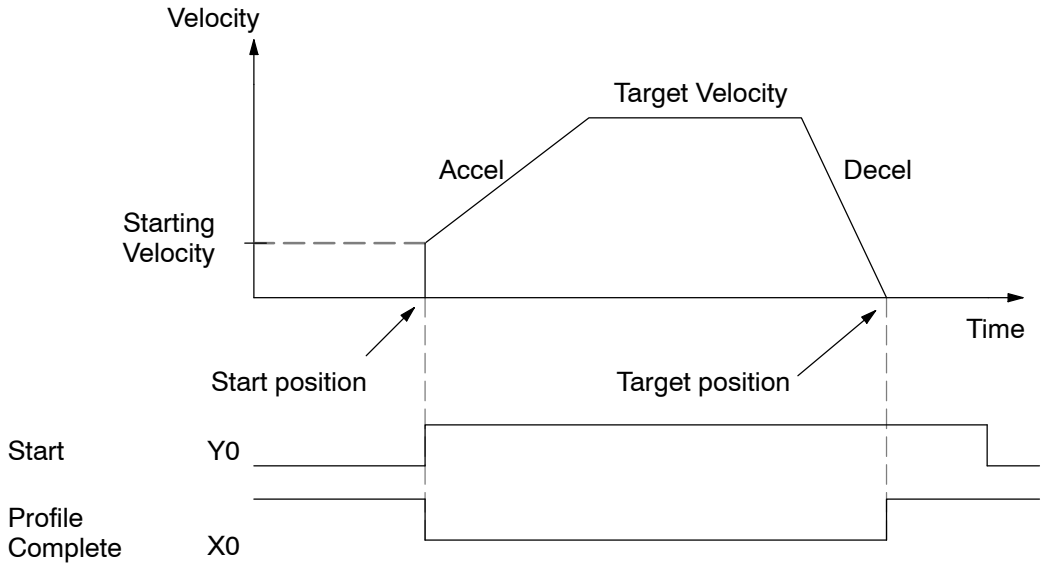


Trapezoidal Profile Operation

Trapezoidal Profile Applications

The trapezoidal profile is best suited for simple point-to-point moves, when the target position is known in advance. Starting velocities must be within the rage of 40 pps to 1 kpps. The remainder of the profile parameters are in the profile parameter table.

Trapezoidal Profile



The time line of signal traces below the profile indicates the order of events.

The HSIO uses logical output Y0 as the Start input to the HSIO, which starts the profile. Immediately the HSIO turns off the Profile Complete signal (logical X0), so the ladder program can monitor the progress of the move. Typically a ladder program will monitor this bit so it knows when to initiate the next profile move.

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

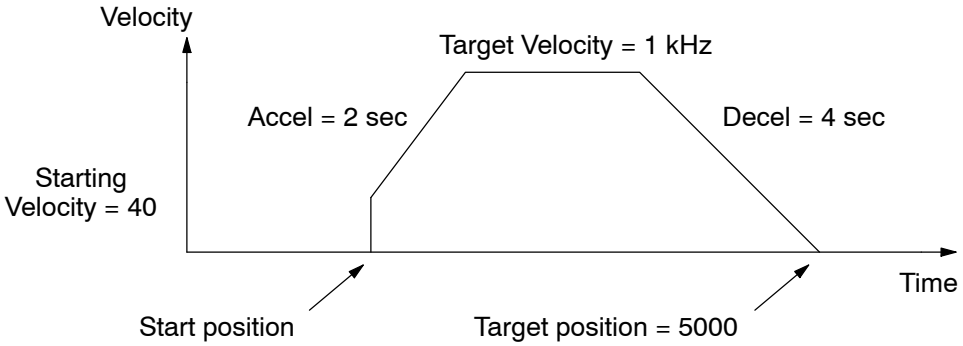
Notice that the motion accelerates immediately to the starting velocity. This segment is useful in stepper systems so we can jump past low speed areas when low-torque problems or a resonant point in the motor might cause a stall. (When a stepper motor stalls, we have lost the position of the load in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

When you need to change the current position value, use logical Y1 output coil to load a new value into the HSIO counter. If the ladder program loads a new value in CT76/CT77 (V1076/V1077), then energizing Y1 will copy that value into the HSIO circuit counter. This must occur before the profile begins, because the HSIO ignores Y1 during motion.

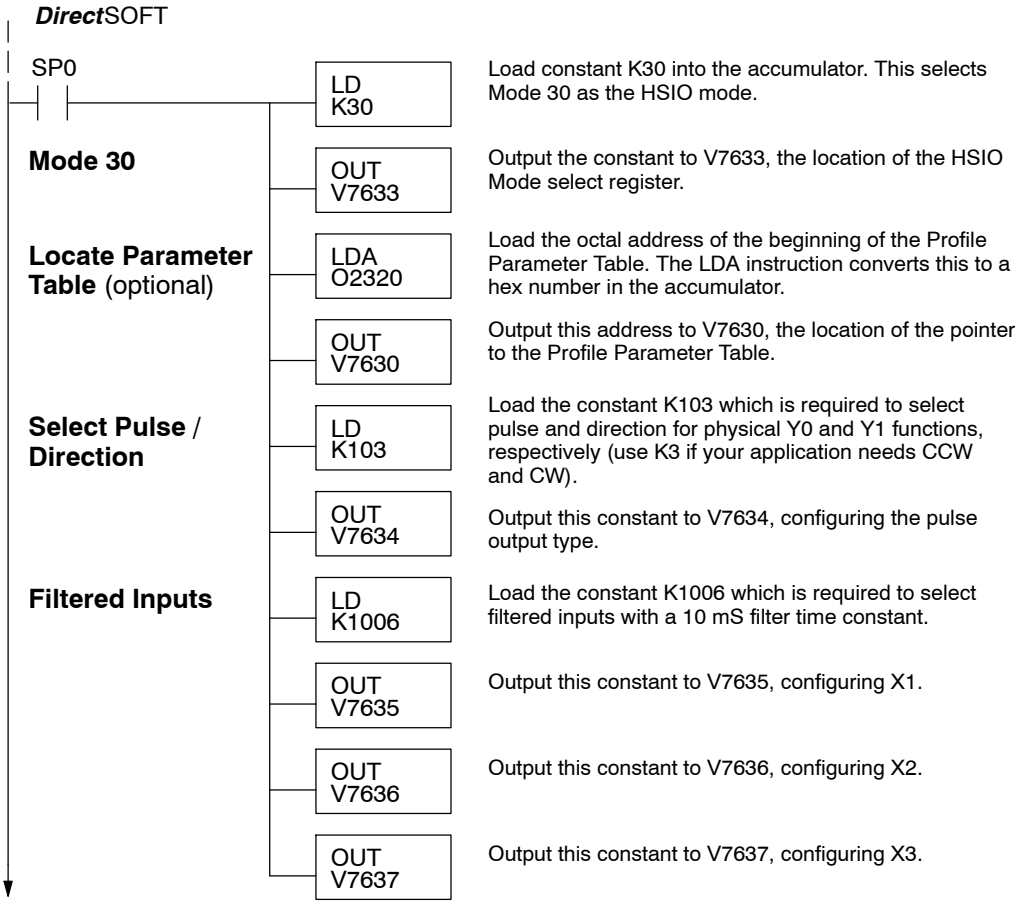
Trapezoidal Profile Program Example

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

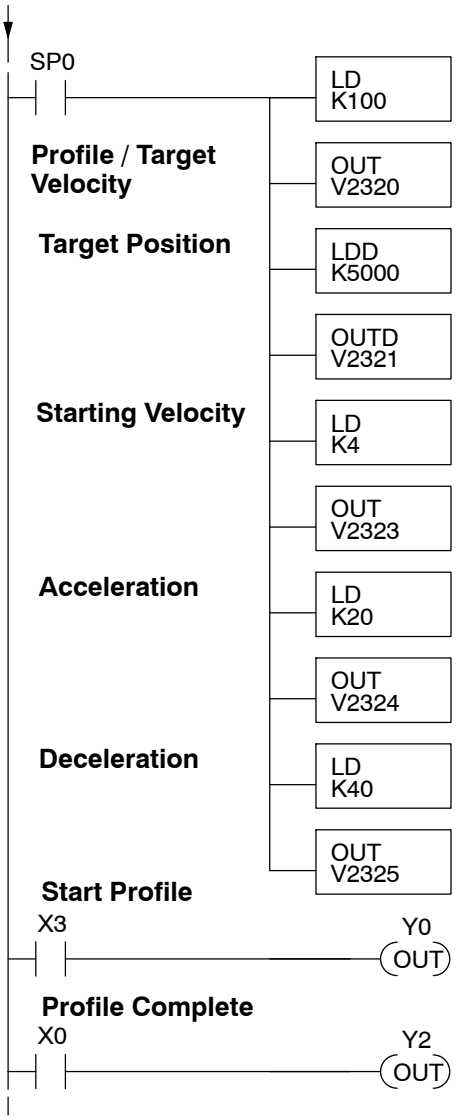
Trapezoidal Profile



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



Program Example Cont'd



Load the constant K100 which is required to select Trapzoidal Profile, absolute positioning, and a target velocity of 1 kHz.

Output this constant to V2320, the location of the Profile Select / Starting Velocity setup register.

Load the constant K5000 which selects a target position of 5000 pulses. Don't forget to use double word size (8-digit BCD position value).

Output this constant to V2321 and V2322, the location of the Target Position double-word register.

Load the constant K4 which is required to select a starting velocity of 40 Hz (4 x 10 pps).

Output this constant to V2323, the location of the starting velocity parameter register.

Load the constant K20 which is required to select an acceleration time of 2 seconds (20 x 100 mS).

Output this constant to V2324, the location of the acceleration parameter register.

Load the constant K40 which is required to select a deceleration time of 4 seconds (40 x 100 mS).

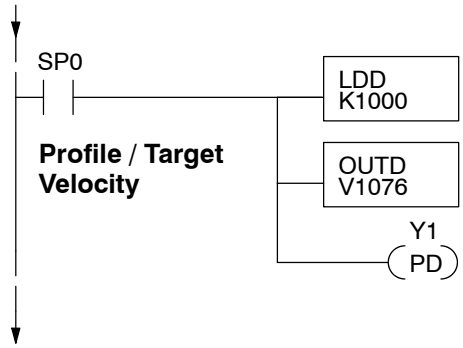
Output this constant to V2325, the location of the deceleration parameter register.

We use a spare filtered input to allow the operator to start the profile. When the operator turns X3 ON, then OFF, logical output Y0 starts the profile.

X0 is the logical output of the HSIO to indicate the move is complete. We use Y2 to energize an annunciator that the profile has finished.

Preload Position Value

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



Load the constant K1000 as the new current position value.

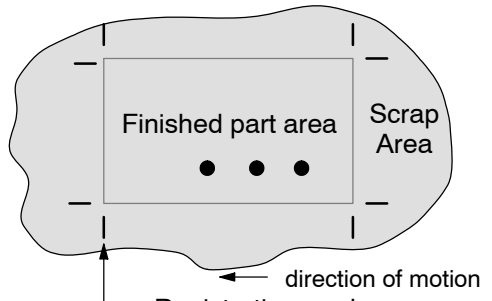
Output this constant to V1076/V1077 (CTA76/CTA77). the location of the current position value.

Turn on Y1 for 1 scan. The off-to-on transition causes the HSIO to preload the current position with the value in

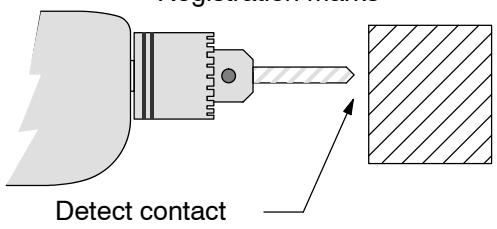
Registration Profile Operation

Registration Applications

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



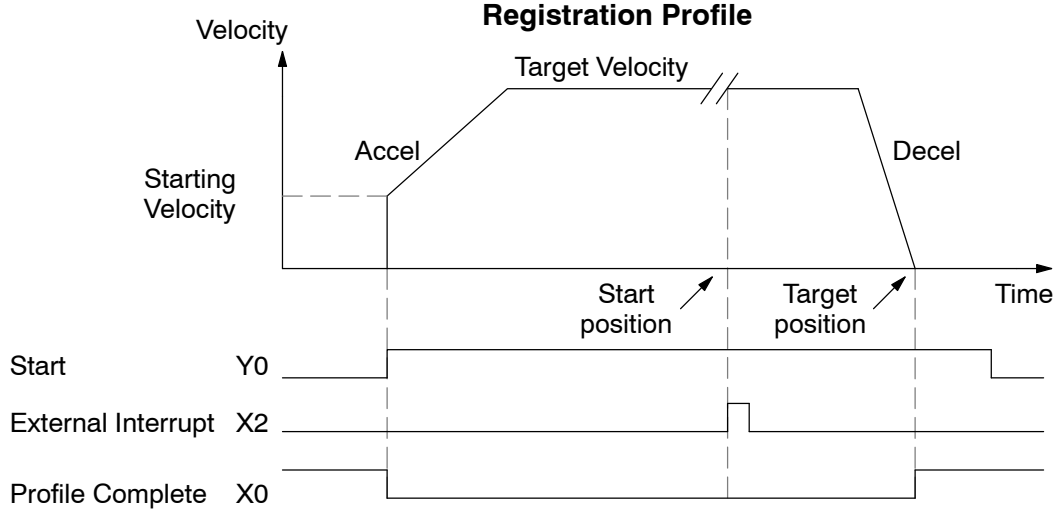
2. In other examples of registration, the work piece is stationary and the tool moves. A drill bit may approach the surface of a part in work, preparing to drill a hole of precise depth. However, the drill bit length gradually decreases due to tool wear. A method to overcome this is to detect the moment of contact with the part surface on each drill, moving the bit into the part a constant distance after contact.



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

The registration profile begins with only velocity control. When an interrupt pulse occurs on physical input X2, the starting position is declared to be the present count (current load position). The velocity control switches to position control, moving the load to the target position. Note that the minimum starting velocity is 40 pps. This instantaneous velocity accommodates stepper motors that can stall at low speeds.

Registration Profile

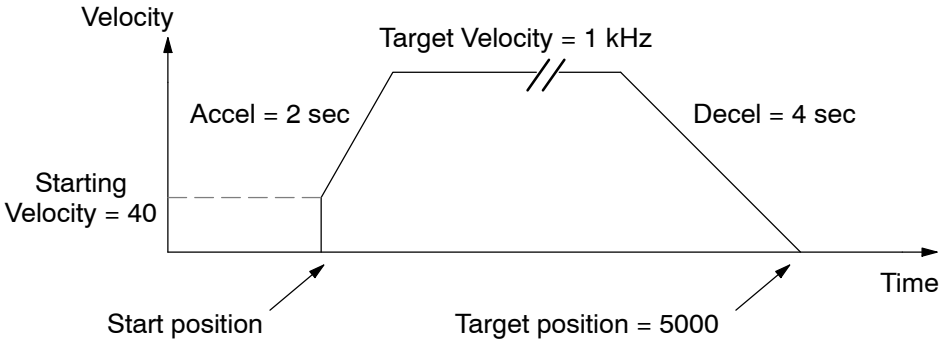


The time line of signal traces below the profile indicates the order of events. The CPU uses logical output Y0 to start the profile. Immediately the HSIO turns off the Profile Complete signal (logical X0), so the ladder program can monitor the move’s completion by sensing the signal’s on state.

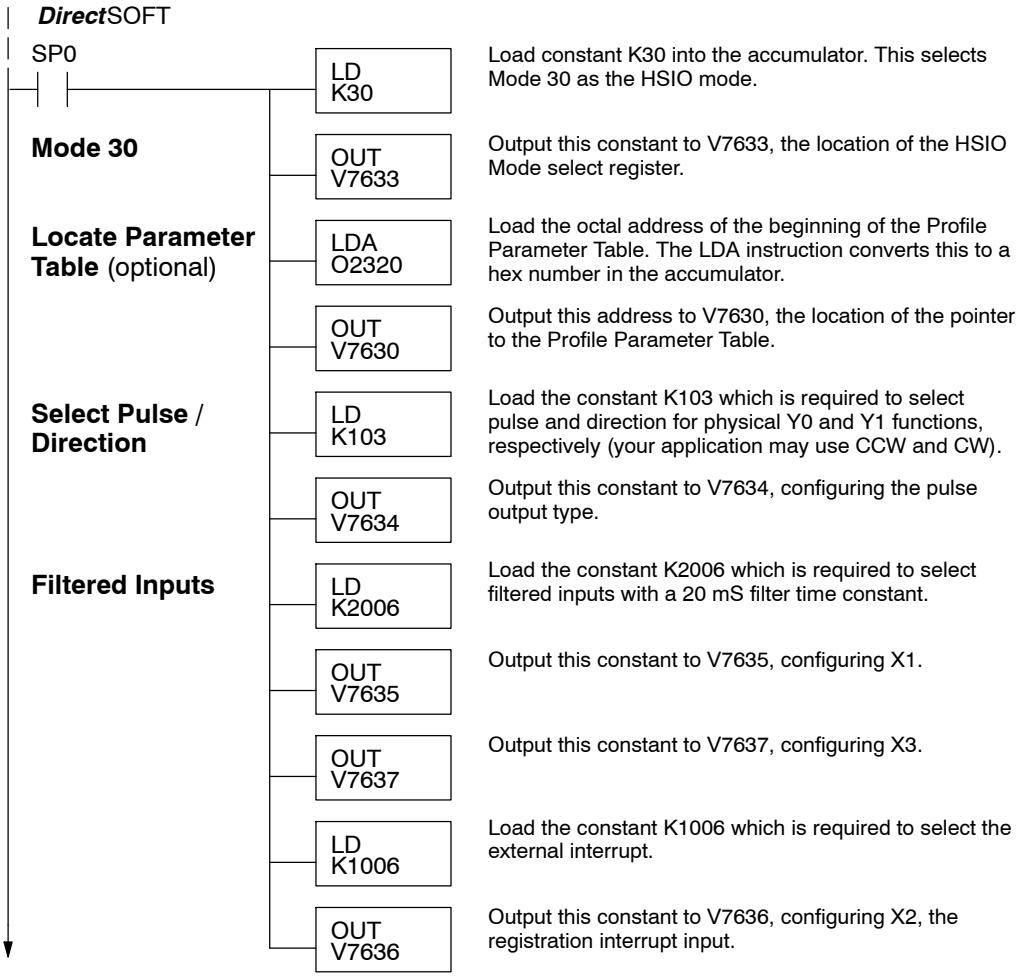
Registration Profile Program Example

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

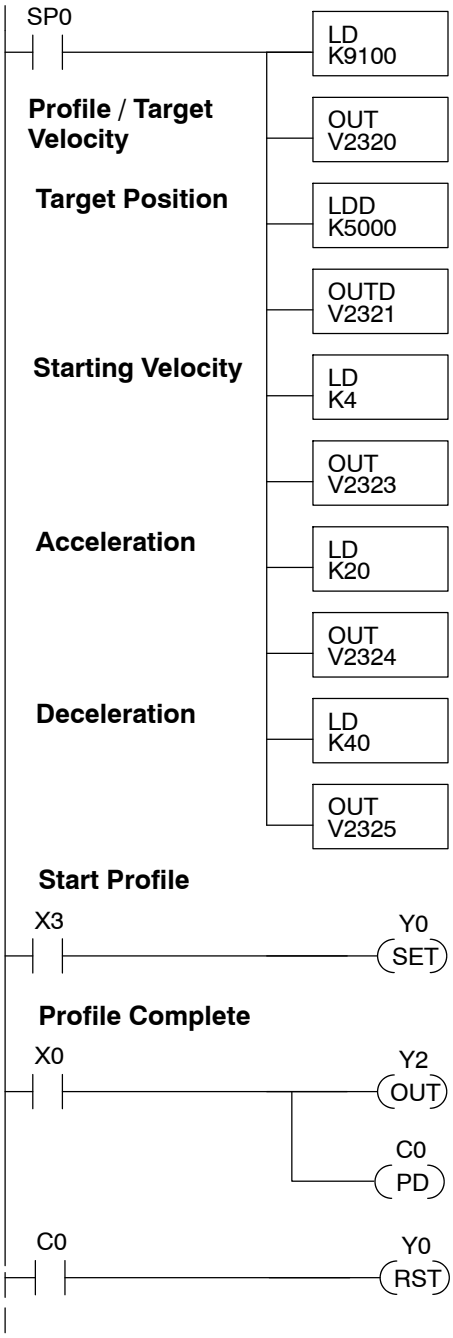
Registration Profile



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



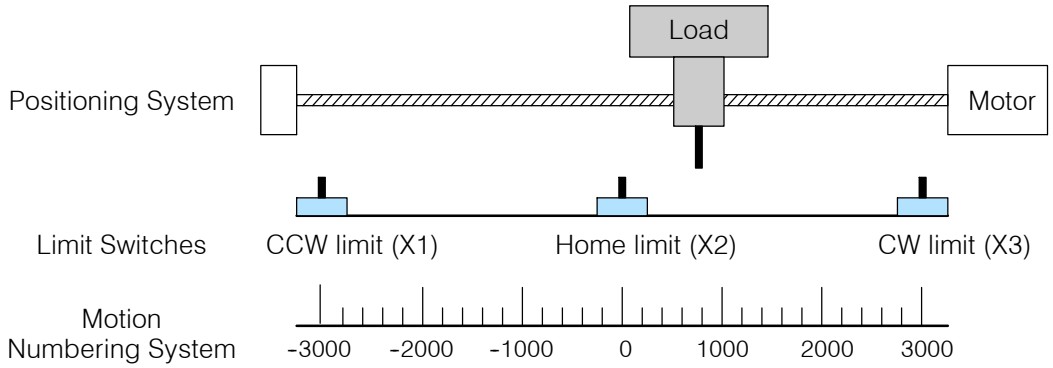
Program Example Cont'd



The profile will begin when the start input (X3) is given. Then the motion begins an indefinite move, which lasts until an external interrupt on X2 occurs. Then the motion continues on for 5000 more pulses before stopping.

Home Search Program Example

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

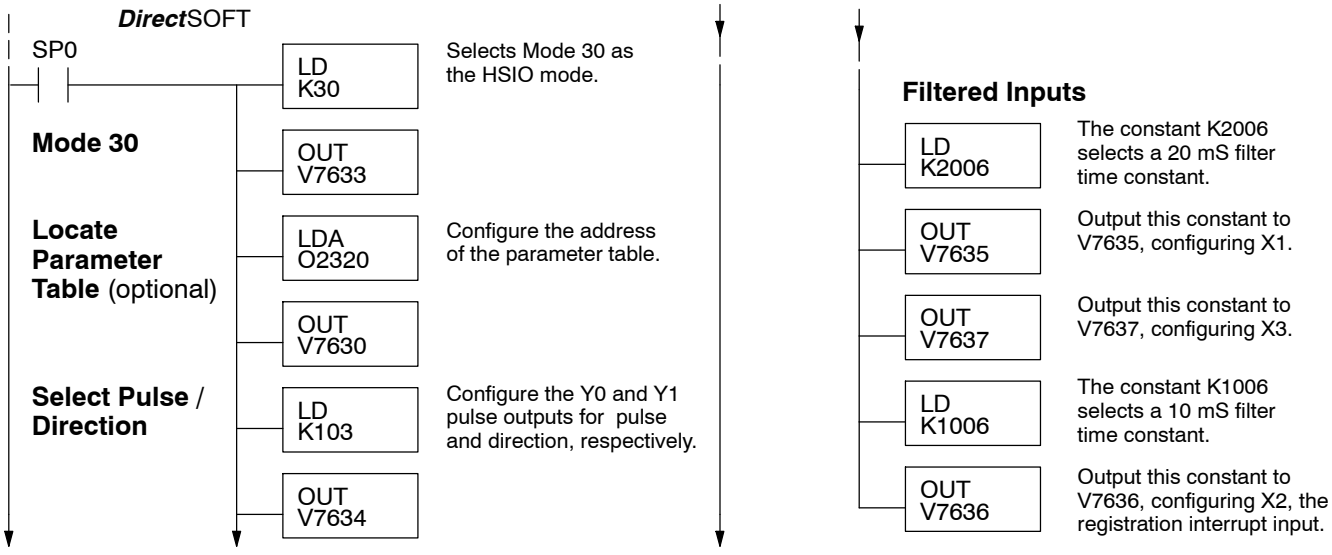


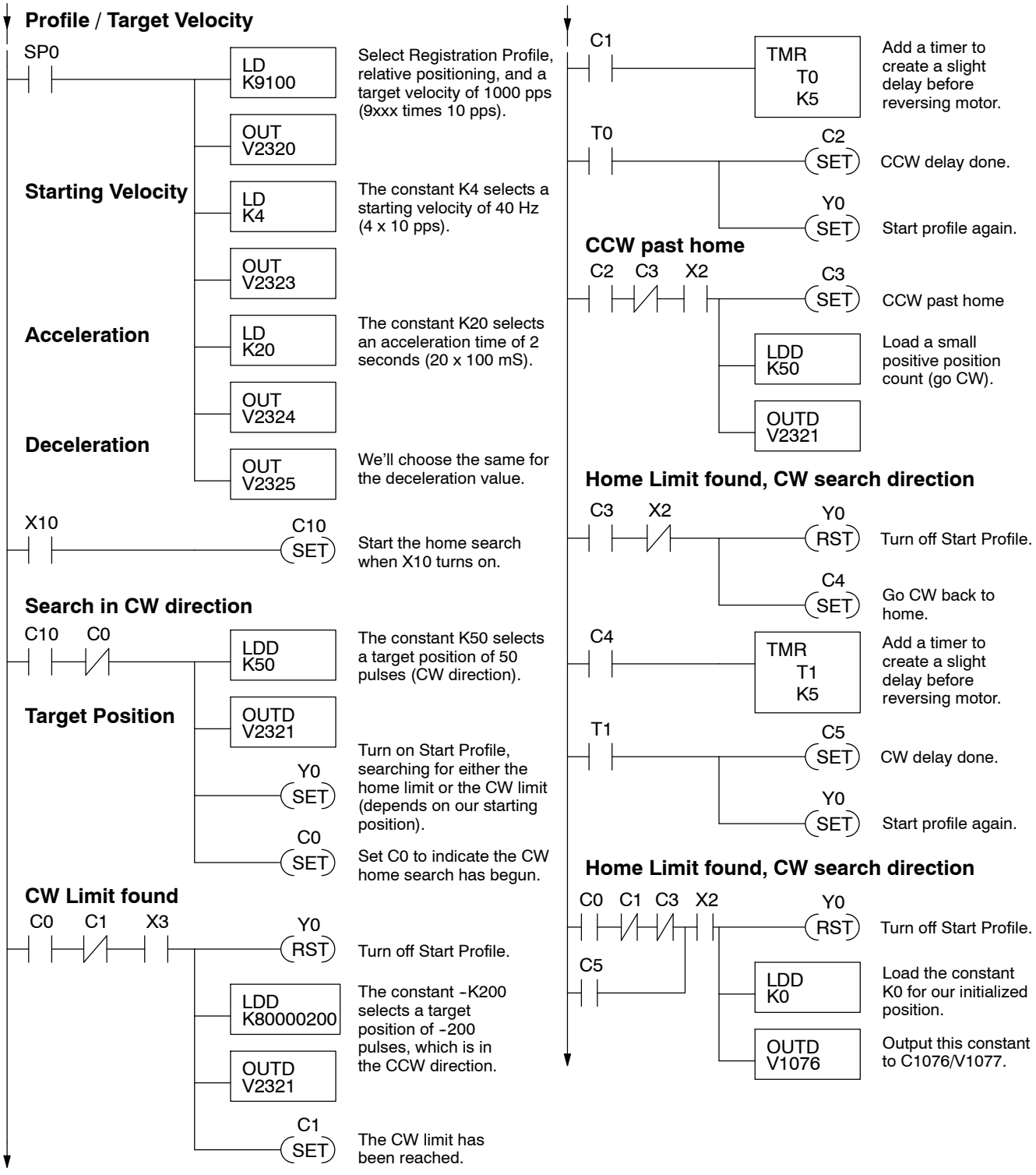
In the drawing above, the load moves left or right depending on the CCW/CW direction of motor rotation. The PLC ladder program senses the CCW and CW limit switches to stop the motor, before the load moves out-of-bounds and damages the machine. The home limit switch is used at powerup to establish the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

At powerup, we do not know whether the load is located to the left or to the right of the home limit switch. Therefore, we will initiate a *home search profile*, using the registration mode. The home limit switch is wired to X2, causing the interrupt. We choose an arbitrary initial search direction, moving in the CW (left-to-right) direction.

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

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





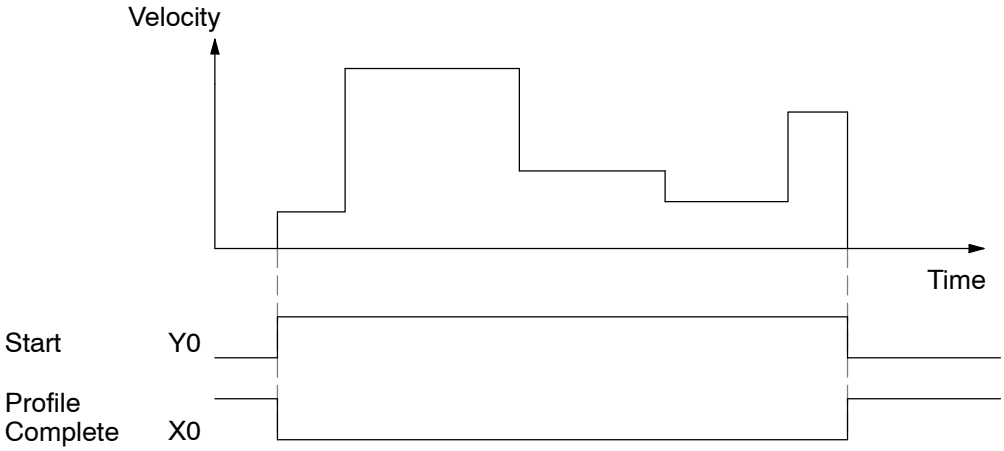
The home search profile will execute specific parts of the program, based on the order of detection of the limit switches. Ladder logic sets C0 to initiate a home search in the CW direction. If the CW limit is encountered, the program searches for home in the CCW direction, passes it slightly, and does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

Velocity Profile Operation

Velocity Profile Applications

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

Velocity Profile



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

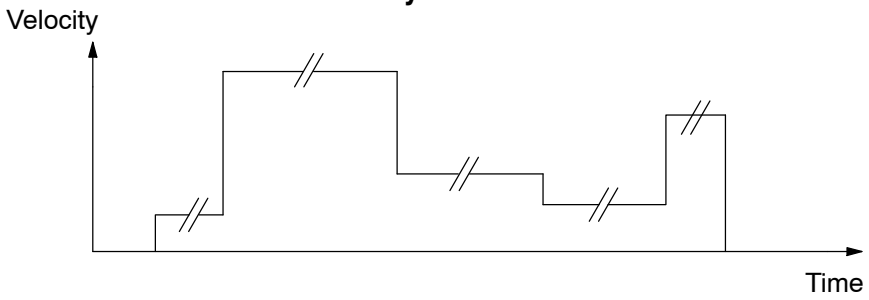
While the Start input is active, the ladder program can command a velocity change by writing a new value to the velocity register (V2323 by default). The full speed range of 40 Hz to 7 kHz is available. Notice from the drawing that there are no acceleration or deceleration ramps between velocity updates. This is how velocity profiling works with the HSIO. However, the ladder program can command more gradual velocity changes by incrementing or decrementing the velocity value more slowly. A counter or timer can be useful in creating your own acceleration/deceleration ramps. Unless the load must do a very complex move, it is easier to let the HSIO function generate the accel/decel ramps by selecting the trapezoidal or registration profiles instead.

Unlike the trapezoidal and registration profiles, you must specify the desired direction of travel with velocity profiles. Load the direction select register (V2321/V2322 by default) with 8000 0000 hex for CCW direction, or 0 for CW direction.

Velocity Profile Program Example

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

Velocity Profile



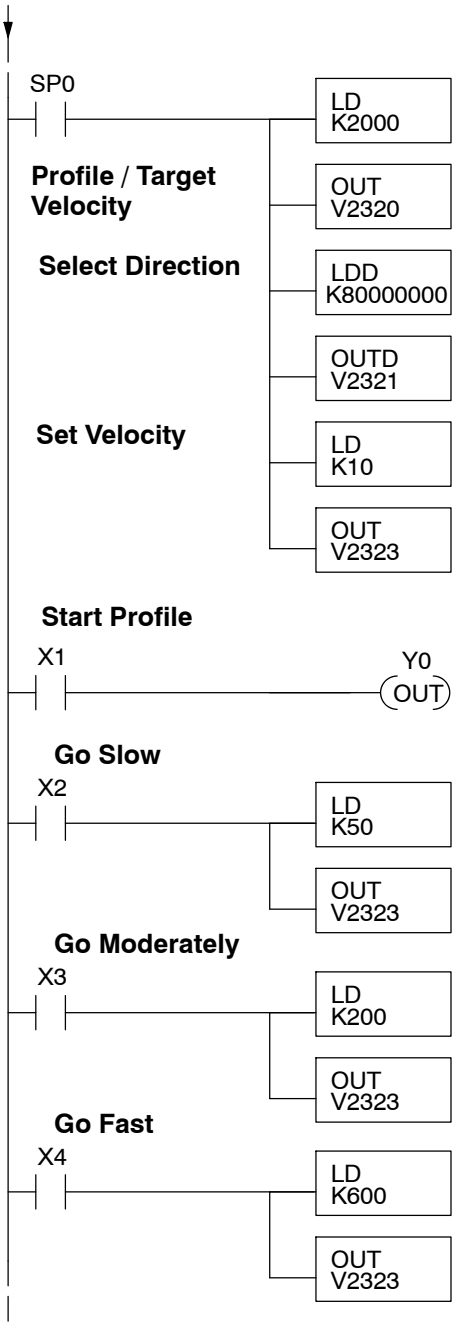
The following program uses dedicated discrete inputs to load in new velocity values. This is a fun program to try, because you can create an infinite variety of profiles with just two or three input switches. The intent is to turn on only one of X1, X2, or X3 at a time. The beginning of the program contains all the necessary setup parameters for Pulse Output Mode 30. We only have to do this once in the program, so we use first-scan contact SP0 to trigger the setup.

DirectSOFT

SP0

	LD K30	Load constant K30 into the accumulator. This selects Mode 30 as the HSIO mode.
Mode 30	OUT V7633	Output this constant to V7633, the location of the HSIO Mode select register.
Locate Parameter Table (optional)	LDA O2320	Load the octal address of the beginning of the Profile Parameter Table. The LDA instruction converts this to hex number in the accumulator.
	OUT V7630	Output this address to V7630, the location of the pointer to the Profile Parameter Table.
Select Pulse / Direction	LD K103	Load the constant K103 which is required to select pulse and direction for physical Y0 and Y1 functions, respectively (your application may use CCW and CW).
	OUT V7634	Output this constant to V7634, configuring the pulse output type.
Filtered Inputs	LD K1006	Load the constant K1006 which is required to select filtered inputs with a 10 mS filter time constant.
	OUT V7635	Output this constant to V7635, configuring X1.
	OUT V7636	Output this constant to V7636, configuring X2.
	OUT V7637	Output this constant to V7637, configuring X3.

Program Example Cont'd



Load the constant K2000 which is required to select Velocity Profile. This data word contains no velocity information in the case of velocity mode.

Output this constant to V2320, the location of the Profile Select setup register.

Load the constant K80000000 which selects CCW direction for Velocity Profiles. Don't forget to use double word size (8-digit BCD position value).

Output this constant to V2321 and V2322, the location of the Target Position double-word register.

Load the constant K10 which is required to select an initial velocity of 100 pps (uses x10 multiplier).

Output this constant to V2323, the location of the velocity parameter register. After the program is running, we can write here again, using discrete input switches.

We use a spare filtered input to allow the operator to start the profile. When the operator turns X1 ON and leaves it on, logical output Y0 starts the profile.

Load the constant K50 which is required to select a velocity of 500 pps when the operator closes X2.

Output this constant to V2323, the location of the velocity parameter register. The speed will change immediately.

Load the constant K200 which is required to select a velocity of 2000 pps when the operator closes X3.

Output this constant to V2323, the location of the velocity parameter register. The speed will change immediately.

Load the constant K600 which is required to select a velocity of 6000 pps when the operator closes X4.

Output this constant to V2323, the location of the velocity parameter register. The speed will change immediately.

Pulse Output Error Codes

The Profile Parameter Table starting at V2320 (default location) defines the profile. Certain numbers will result in a error when the HSIO attempts to use the parameters to execute a move profile. When an error occurs, the HSIO writes an error code in V2326.

Error Code	Error Description
0000	No error
0010	Requested profile type code is invalid (must use 0, 1, 2, 8, or 9)
0020	Target Velocity is not in BCD
0021	Target Velocity is specified to be less than 40 pps
0022	Target Velocity is specified to be greater than 7,000 pps
0030	Target Position value is not in BCD
0040	Starting Velocity is not in BCD
0041	Starting Velocity is specified to be less than 40 pps
0042	Starting Velocity is specified to be greater than 1,000 pps
0050	Acceleration Time is not in BCD
0051	Acceleration Time is zero
0052	Acceleration Time is greater than 10 seconds
0010	Deceleration Time is not in BCD
0010	Deceleration Time is zero
0010	Deceleration Time is greater than 10 seconds

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

Troubleshooting Guide for Mode 30

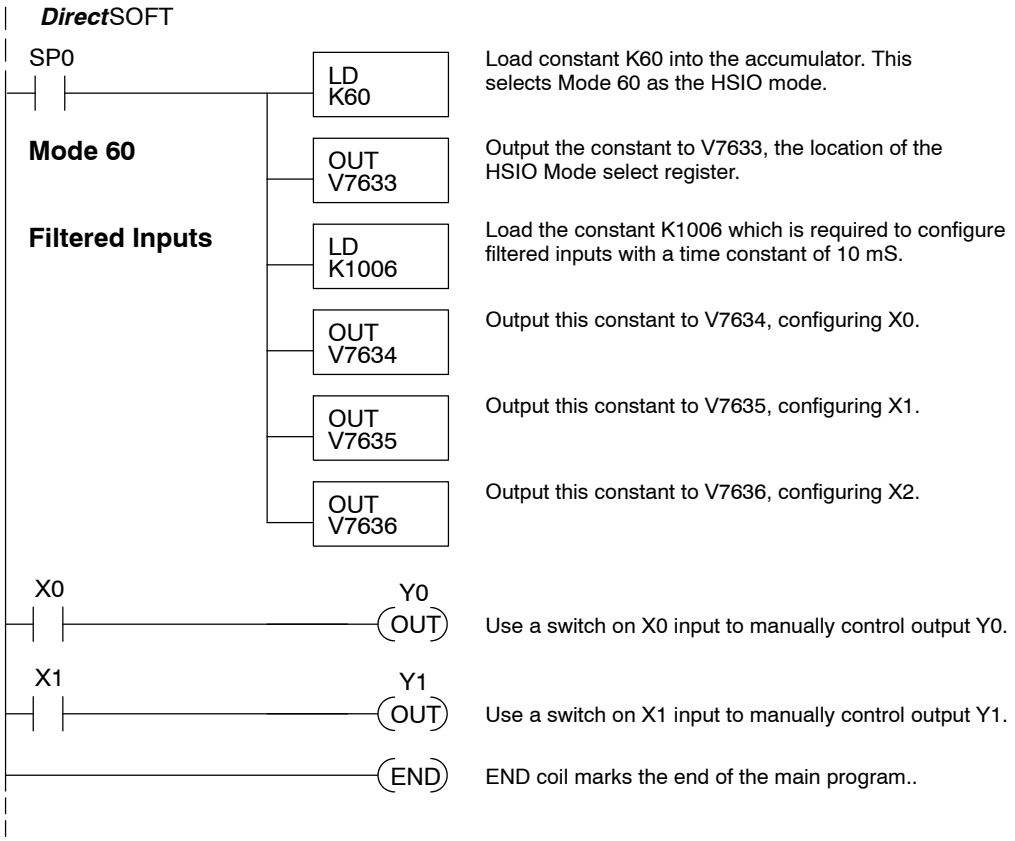
If you're having trouble with Mode 30 operation, please study the following symptoms and possible causes. The most common problems are listed below:

Symptom: The stepper motor does not rotate.

Possible causes:

1. **Configuration** – Verify that the HSIO actually generates pulses on outputs Y0 and Y1. Watch the status LEDs for Y0 and Y1 when you start a motion profile. If the LEDs flicker on and off or are steadily on, the configuration is probably correct.
2. **Programming error** – If there are no pulses on Y0 or Y1 you may have a programming error. Check the contents of V2326 for an error code that may be generated when the PLC attempts to do the move profile. Error code descriptions are given above.

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



5. **Memory Error** – HSIO configuration parameters are stored in the CPU system memory. Corrupted data in this memory area can sometimes interfere with proper HSIO operation. If all other corrective actions fail, initializing the scratchpad memory may solve the problem. With ***Direct***SOFT5, select the *PLC* menu, then *Setup*, then *Initialize Scratchpad*.

Symptom: The motor turns in the wrong direction.

Possible causes:

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

Mode 40: High-Speed Interrupts

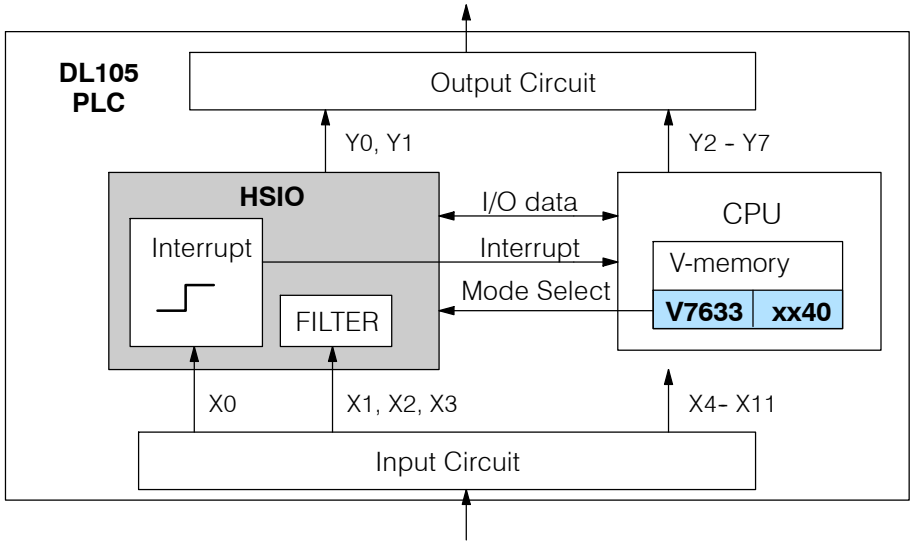
Purpose

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

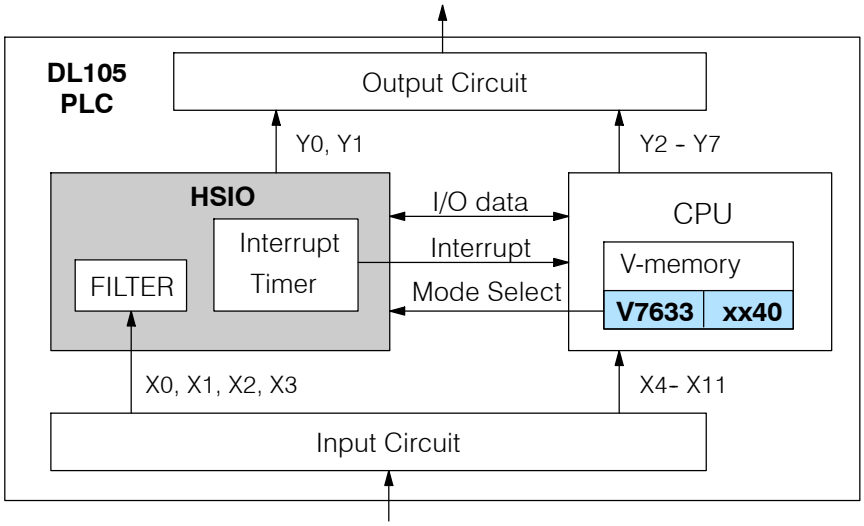
- An external event needs to trigger an interrupt subroutine in the CPU. Using immediate I/O instructions in the subroutine is typical.
- An interrupt routine needs to occur on a timed basis which is different from the CPU scan time (either faster or slower). The timed interrupt is programmable, from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

Functional Block Diagram

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



Alternately, you may configure the HSIO circuit to generate interrupts based on a timer, as shown below. In this configuration, inputs X0 through X3 are filtered inputs.



Setup for Mode 40

Recall that V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 40 in the lower byte to select High-Speed Interrupt Mode. Use BCD 00 or 20 in the upper byte as required. Combine the two bytes into a data word "xx40", for writing to V7633.

Memory Location V7633

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0

2 0 — Miscellaneous Setup (BCD)
00 = Power Up in Previous Mode
20 = Power Up in Run Mode Always

4 0 — HSIO Mode Setup (BCD)
40 = High-Speed Interrupt

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

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

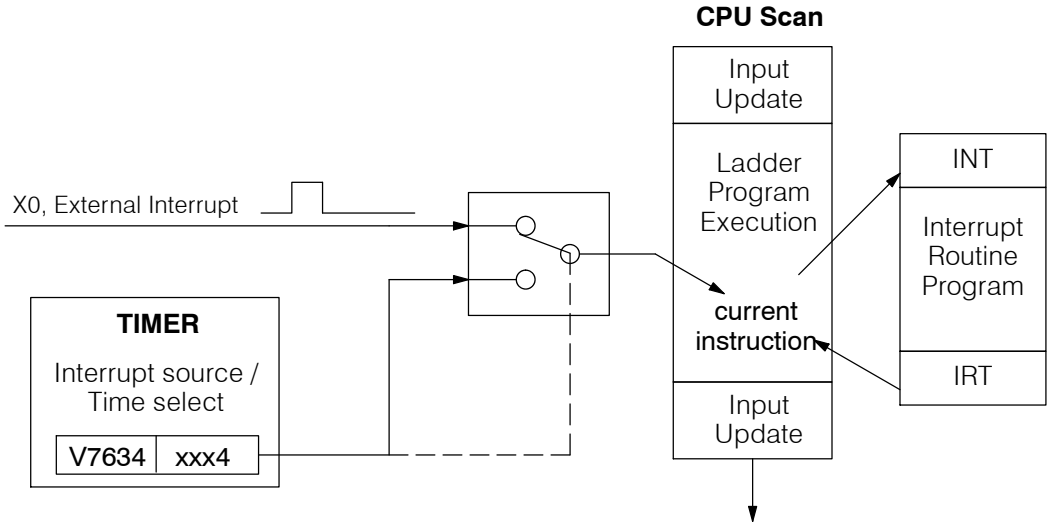
We recommend using the first method above so that the HSIO setup becomes an integral part of your application program. An example program later in this section shows how to to this.

Interrupts and the Ladder Program

Refer to the drawing below. The source of the interrupt may be external (X0), or the HSIO timer function. The setup parameter in V7634 serves a dual purpose:,

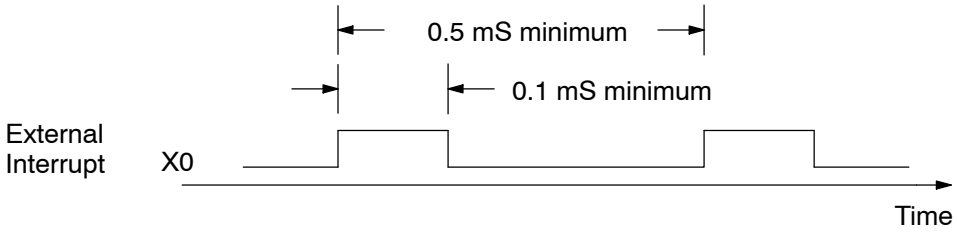
- It selects between the two interrupt sources, external (X0) or an internal timer.
- In the case of the timer interrupt, it programs the interrupt timebase between 5 and 999 mS.

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



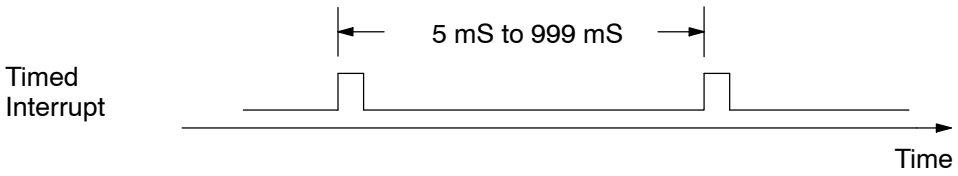
External Interrupt Timing Parameters

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



Timed Interrupt Parameters

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



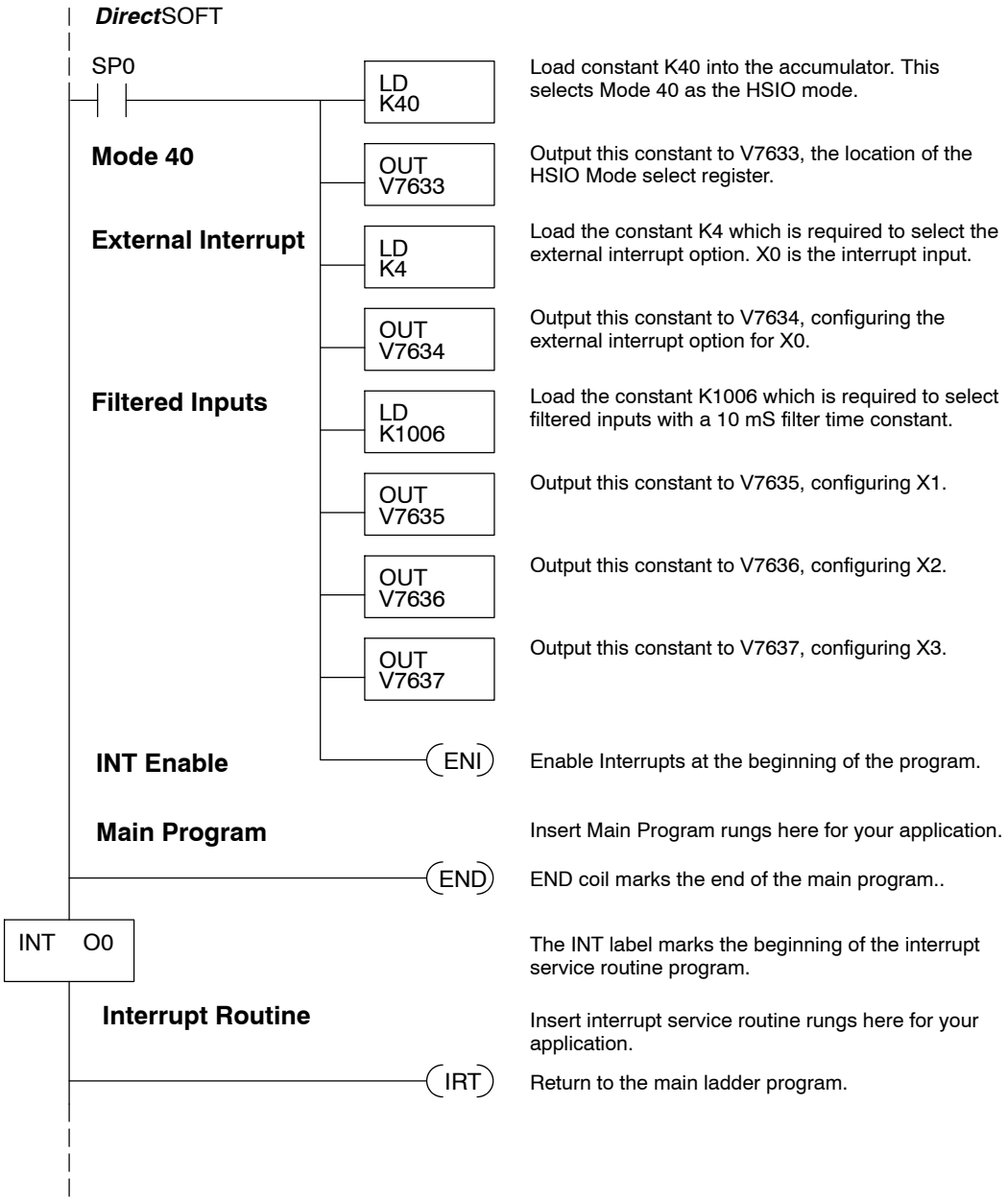
X Input / Timed INT Configuration

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

Input	Configuration Register	Function	Hex Code Required
X0	V7634	External Interrupt	0004
	Uses X1's code in V7635	Filtered Input (when timed interrupt is in use)	0054 to 9994, which is the timed INT timebase
X1	V7635	Filtered Input	xx06 (xx = filter time)
X2	V7636	Filtered Input	xx06 (xx = filter time)
X3	V7637	Filtered Input	xx06 (xx = filter time)

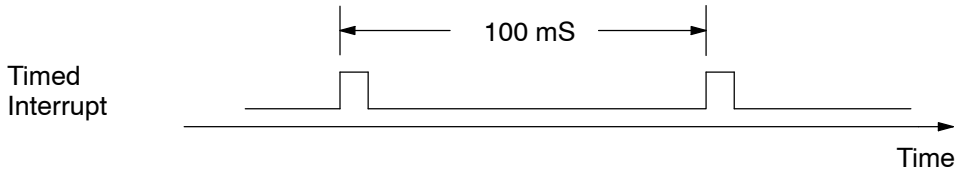
External Interrupt Program Example

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

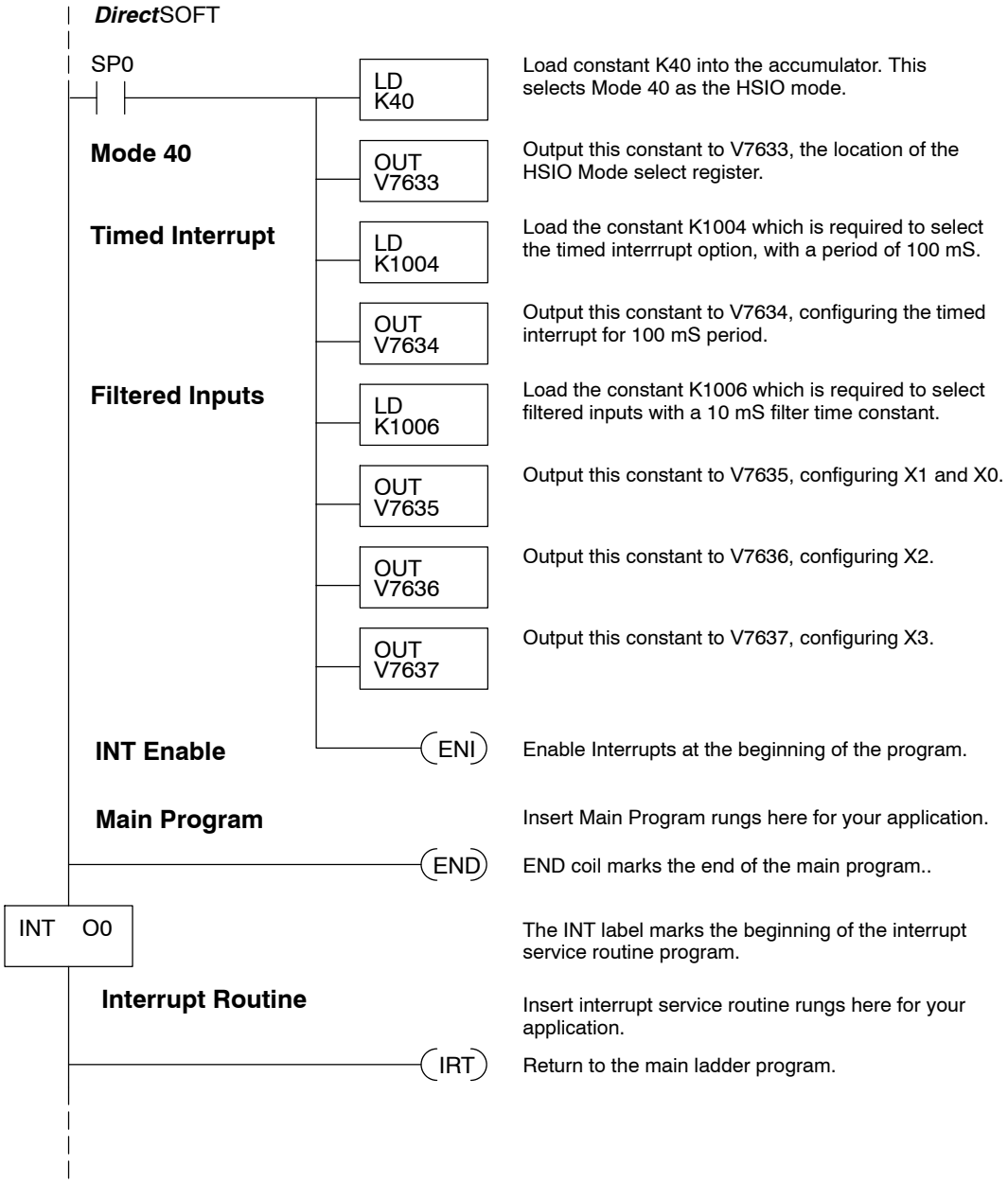


Timed Interrupt Program Example

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



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



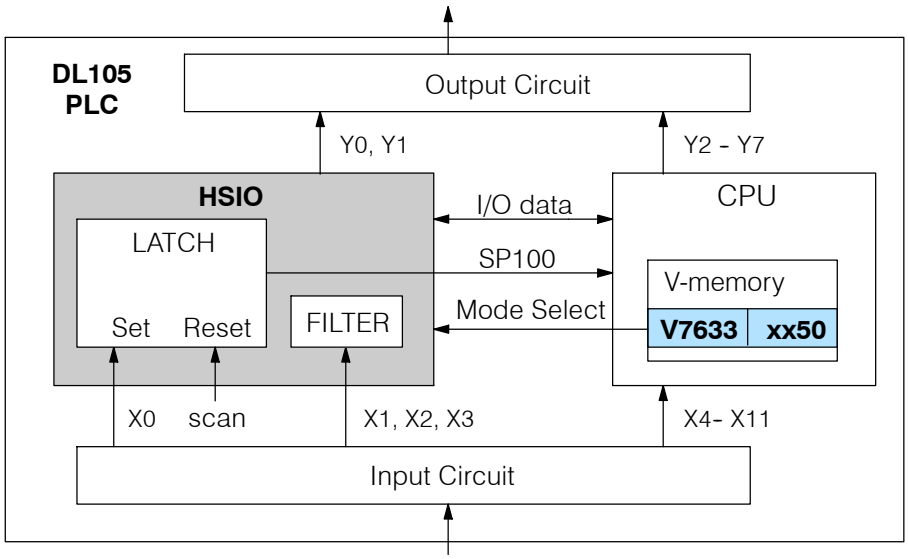
Mode 50: Pulse Catch Input

Purpose

The HSIO circuit has a pulse-catch mode of operation. It monitors the signal on input X0, preserving the occurrence of a narrow pulse. The purpose of the pulse catch mode is to enable the ladder program to "see" an input pulse which is shorter in duration than the current scan time. The HSIO circuit latches the input event on input X0 for one scan, and presents it to ladder logic through special relay SP100 contact. This contact automatically goes off after one scan. *Note that the ladder program cannot read the status of X0 directly.*

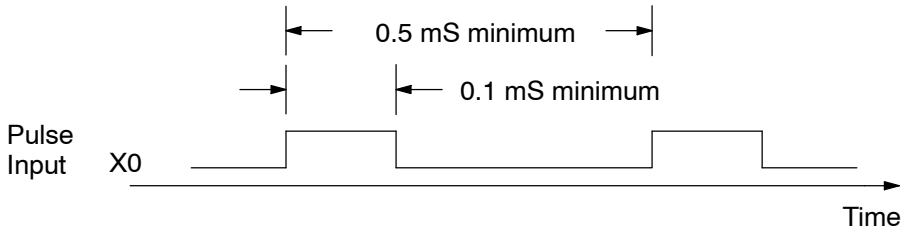
Functional Block Diagram

Refer to the block diagram below. When the lower byte of HSIO Mode register V7633 contains a BCD "50", the pulse catch mode in the HSIO circuit is enabled. X0 automatically becomes the pulse catch input, which sets the latch on each rising edge. The HSIO resets the latch at the end of the next CPU scan. The latch output is available to the ladder program through the special relay contact SP100. Inputs X1, X2, and X3 are available as filtered discrete inputs.



Pulse Catch Timing Parameters

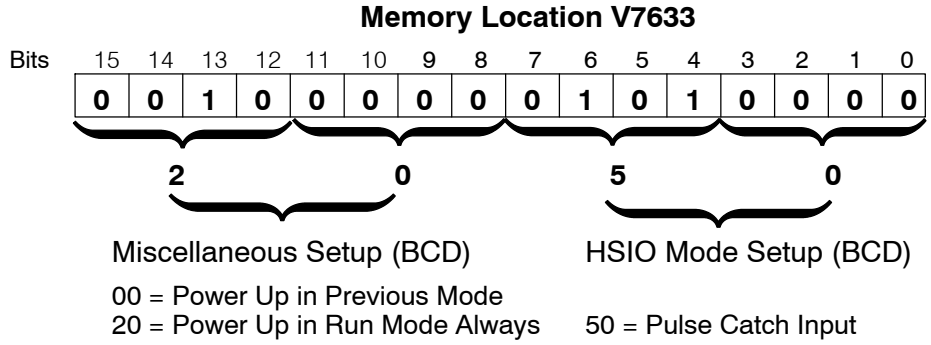
Signal pulses at X0 must meet certain timing criteria to guarantee a pulse capture will result. Refer to the timing diagram below. The input characteristics of X0 are fixed (it is not a programmable filtered input). The minimum pulse width is 0.1 mS. There must be some delay before the next pulse arrives,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



Note that the pulse catch and filtered input functions are opposite in nature. The pulse catch feature on X0 seeks to *capture* narrow pulses, while the filter input feature on X1, X2, and X3 seeks to *reject* narrow pulses.

Setup for Mode 50

Recall that V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 50 in the lower byte to select High-Speed Counter Mode. Use BCD 00 or 20 in the upper byte as required. Combine the two bytes into a data word "xx50", for writing to V7633.



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

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

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

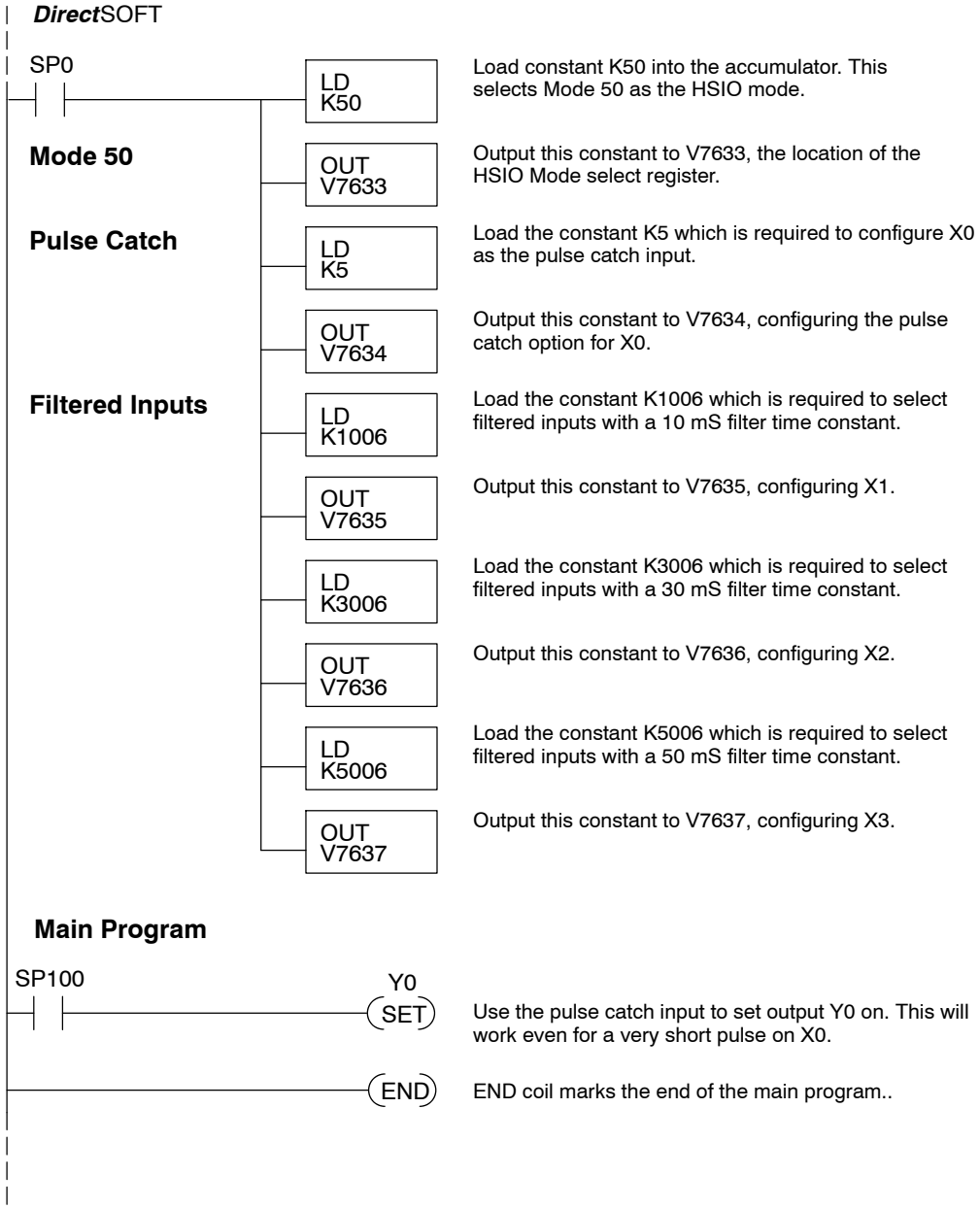
X Input Configuration

The configurable discrete input options for Pulse Catch Mode are listed in the table below. Input X0 is the pulse input, and must have "0005" loaded into it configuration register V7634. Inputs X1, X2, and X3 can only be filtered inputs. Each input has its own configuration register and filter time constant.

Input	Configuration Register	Function	Hex Code Required
X0	V7634	Pulse Catch Input	0005
X1	V7635	Filtered Input	xx06 (xx = filter time)
X2	V7636	Filtered Input	xx06 (xx = filter time)
X3	V7637	Filtered Input	xx06 (xx = filter time)

Pulse Catch Program Example

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



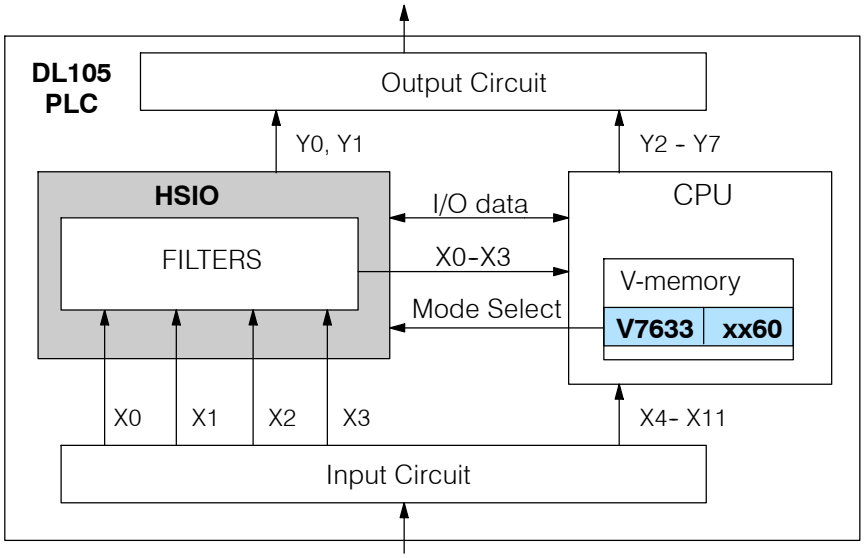
Mode 60: Discrete Inputs with Filter

Purpose

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

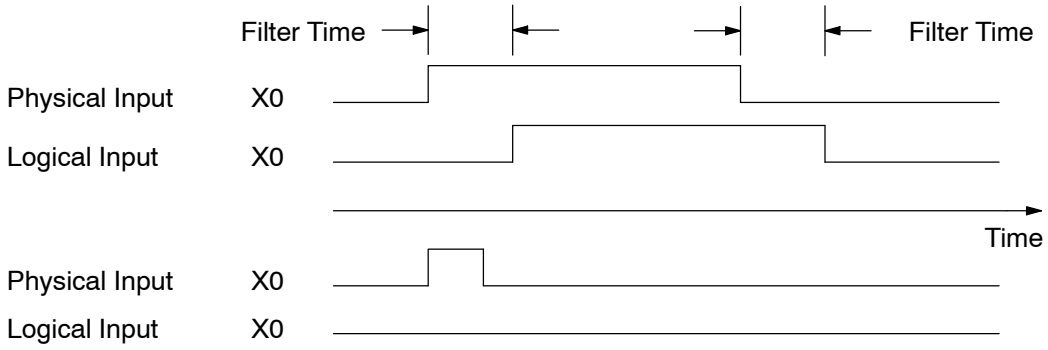
Functional Block Diagram

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



Input Filter Timing Parameters

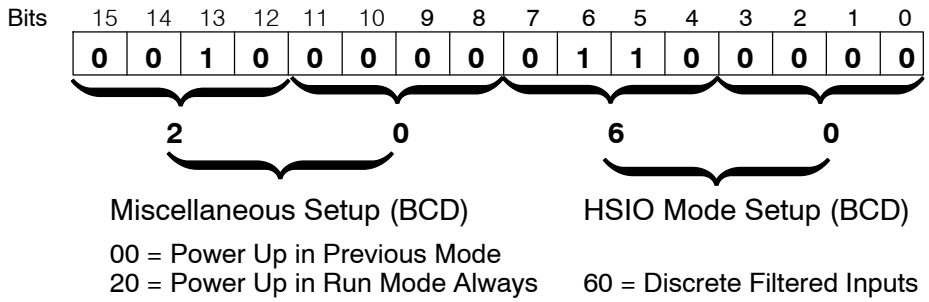
Signal pulses at inputs X0 – X3 are filtered by using a delay time. In the figure below, the input pulse on the top line is longer than the filter time. The resultant logical input to ladder is phase-shifted (delayed) by the filter time on both rising and falling edges. In the bottom waveforms, the physical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



Setup for Mode 60

Recall that V7633 is the HSIO Mode Select register. Refer to the diagram below. Use BCD 60 in the lower byte to select High-Speed Counter Mode. Use BCD 00 or 20 in the upper byte as required. Combine the two bytes into a data word "xx60", for writing to V7633.

Memory Location V7633



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

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

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

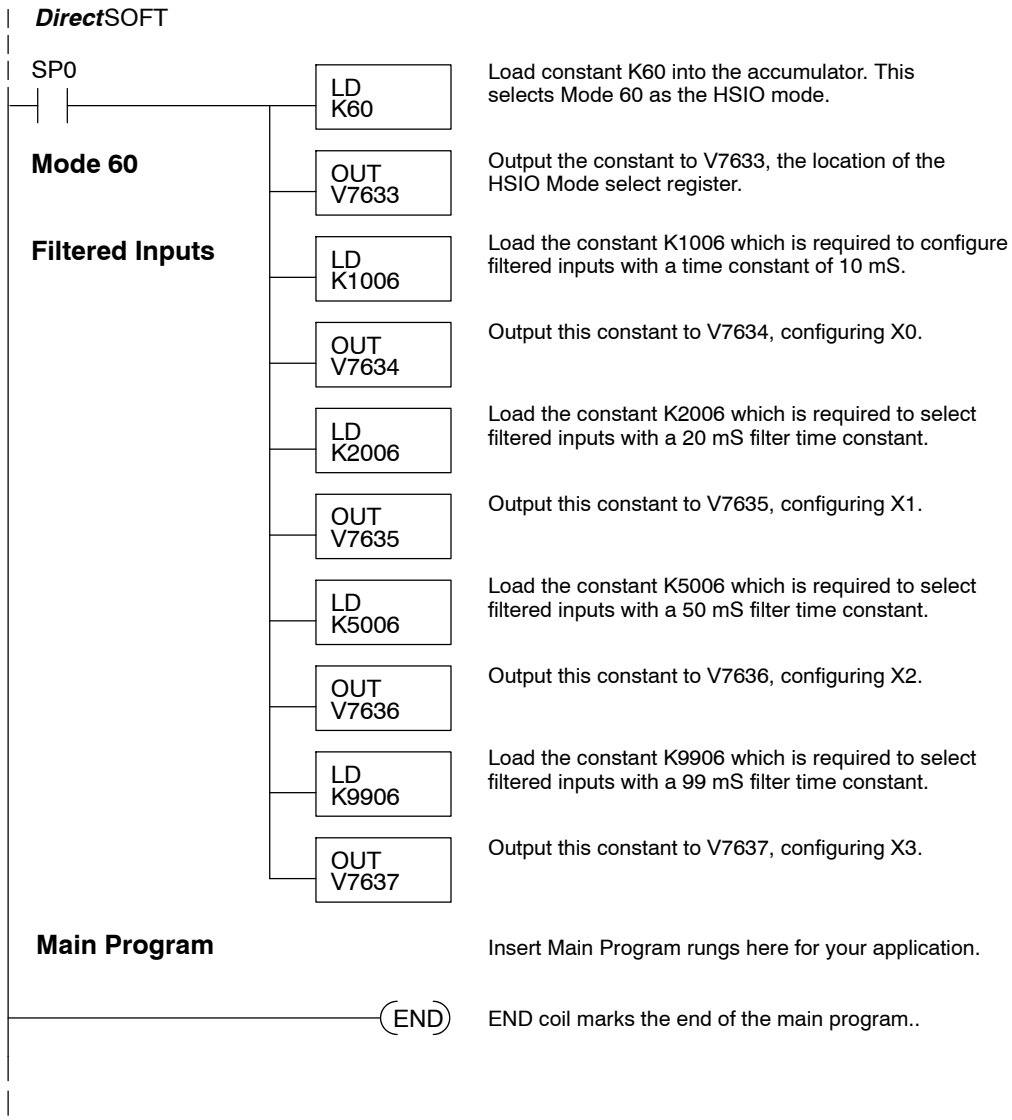
X Input Configuration

The configurable discrete input options for Discrete Filtered Inputs Mode are listed in the table below. The filter time constant (delay) is programmable from 10 to 99 mS. The code for this selection occupies the upper byte of the configuration register in BCD. We combine this number with the required "06" in the lower byte to get "xx06", where xx = 10 to 99. Input X0, X1, X2, and X3 can only be filtered inputs. Each input has its own configuration register and filter time constant.

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

Filtered Inputs Program Example

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



4

CPU Specifications and Operation

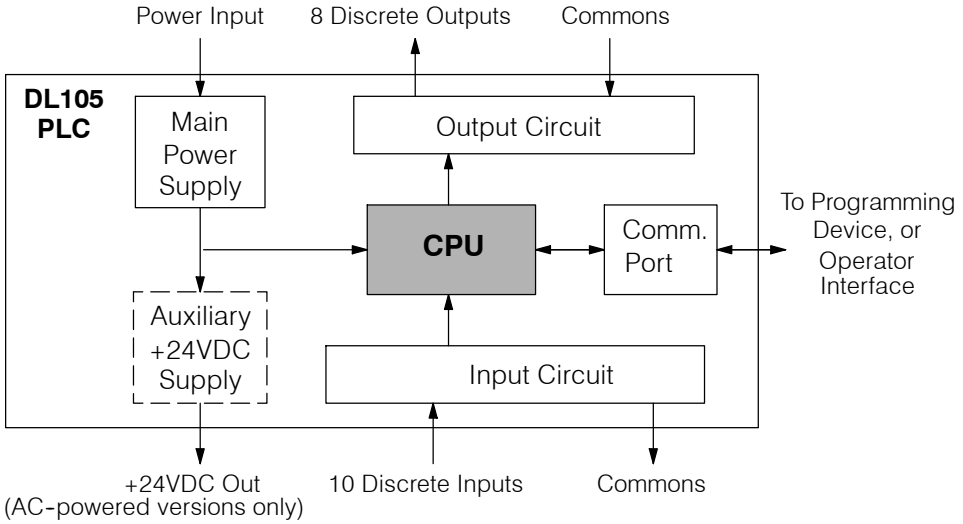
In This Chapter. . . .

— Introduction
— CPU Specifications
— CPU Hardware Setup
— CPU Operation
— Program Mode Operation
— Run Mode Operation
— I/O Response Time
— CPU Scan Time Considerations
— PLC Numbering Systems
— Memory Map
— DL105 System V-Memory
— X Input Bit Map
— Y Output Bit Map
— Control Relay Bit Map
— Stage Control / Status Bit Map
— Timer Status Bit Map
— Counter Status Bit Map

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 program, organization of Variable Memory





NOTE: The High-Speed I/O function (HSIO) consists of dedicated but configurable hardware in the DL105. It is not considered part of the CPU, because it does not execute the ladder program. For more on HSIO operation, see Chapter 3.

DL105 CPU Features

The DL105 Micro PLC which has 2.4K words of memory comprised of 2.0K of ladder memory and 384 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 days in the event of a power outage. The capacitor automatically charges during powered operation of the PLC.

The DL105 supports fixed I/O which includes ten discrete input points and eight output points. No provision for expansion beyond these eighteen I/O points are available in the F1-130 model PLCs.

Over 90 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. Chapter 5 provides a detailed description of the instructions.

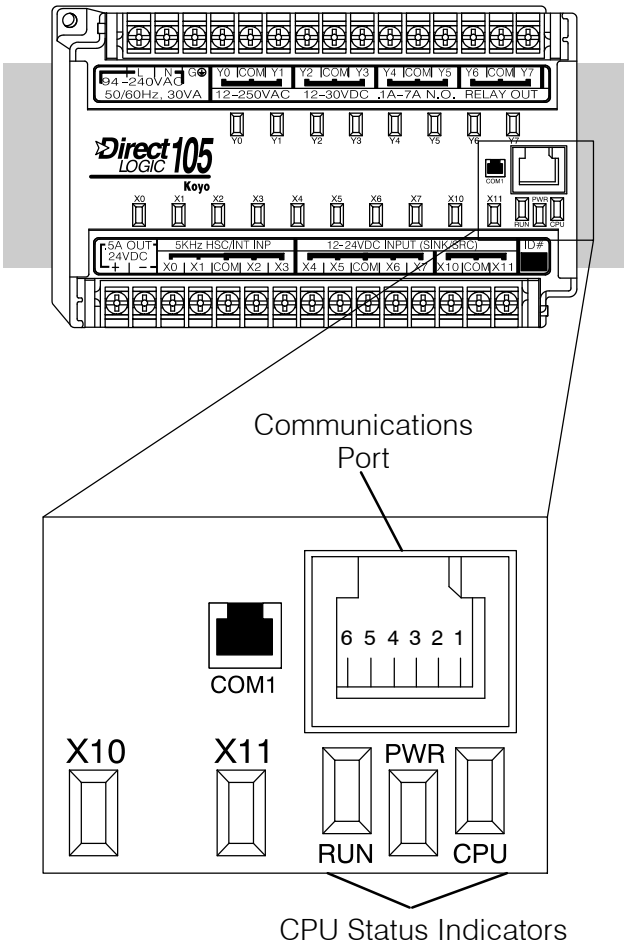
The DL105 provides one built-in RS232C communication port, so you can easily connect a handheld programmer or a personal computer without needing any additional hardware.

CPU Specifications

Feature	DL105
Total Program memory (words)	2.4K
Ladder memory (words)	2048
Total V-memory (words) (See Appendix E)	384
User V-memory (words)	256
Non-volatile V-Memory (words)	128
Contact execution (boolean)	3.3 μS
Typical scan (boolean)	4 – 6 mS
RLL Ladder style Programming	Yes
RLL and RLL*PLUS* Programming	Yes
Run Time Edits	Yes
Variable / fixed scan	Variable
Supports Override	No
Handheld programmer	Yes
DirectSOFT programming for Windows™	Yes
Built-in communication ports (RS232C)	Yes
EEPROM or FLASH	Standard on CPU
Local Discrete I/O points available	18
Local Analog input / output channels maximum	None
High-Speed I/O (quad., pulse out, interrupt, pulse catch, etc.)	Yes
I/O Point Density	10 inputs, 8 outputs
Number of instructions available (see Chapter 5 for details)	91
Control relays	256
Special relays (system defined)	112
Stages in RLL*PLUS*	256
Timers	64
Counters	64
Immediate I/O	Yes
Interrupt input (external / timed)	Yes
Subroutines	No
For/Next Loops	No
Math	Integer
Drum Sequencer Instruction	Yes
Time of Day Clock/Calendar	No
Internal diagnostics	Yes
Password security	Yes
System error log	No
User error log	No
Battery backup	No (uses super-cap.)

CPU Hardware Setup

CPU Status Indicators		
RUN	ON	CPU is in RUN mode
	OFF	CPU is in Program mode
	FLASHING	CPU is in Upgrade mode
CPU	ON	CPU internal diagnostics has detected an error.
	OFF	CPU is OK.
PWR	ON	CPU power good
	OFF	CPU power failure

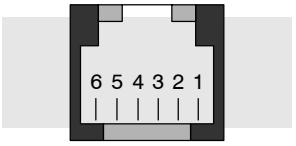


Communication Port	
Com 1	Connects to HPP, ***Direct***SOFT, operator interfaces, etc. 6-pin, RS232C 9600 Baud Odd parity Station address fixed (1) 8 data bits 1 start, 1 stop bit Asynchronous, Half-duplex, DTE K sequence protocol

Communication Port Pinout Diagrams

Cables are available that allow you to quickly and easily connect a Handheld Programmer or a personal computer to the DL105 PLCs. However, if you need to build your own cables, use the pinout diagrams shown. The DL105 PLCs require an RJ-12 phone plug to fit the built-in jacks.

The Micro PLC has one built-in RS232C communication port. The port is generally used for programming either with the Handheld Programmer or ***Direct***SOFT, and has a fixed station address of 1. The baud rate is fixed at 9600 baud. This port supports the K-sequence protocol, which is a proprietary protocol.



Phone Jack Connector

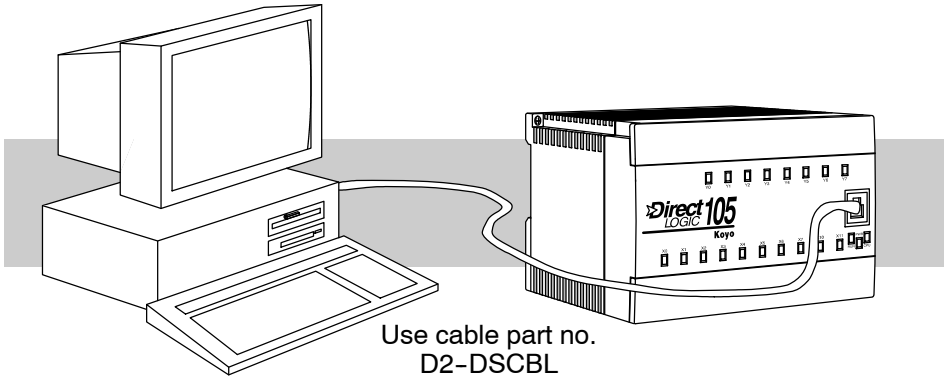
Port Pinouts

Pin	Signal Definition
1	0 V
2	5 V
3	RS232C Data in
4	RS232C Data out
5	5 V

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

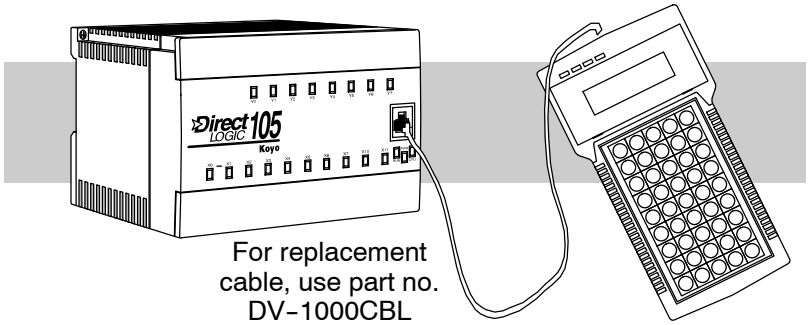
Connecting the Programming Devices

If you're using a Personal Computer with the ***Direct***SOFT programming package, you can connect the computer to the DL105's programming port. For an engineering office environment (typical during program development), this is the preferred method of programming.



Use cable part no.
D2-DSCBL

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



For replacement
cable, use part no.
DV-1000CBL

CPU Setup Information

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

- Using Auxiliary Functions
- Selecting and Changing the CPU Modes
- 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 ***Direct***SOFT Manual provides a description of the menus and keystrokes required to perform the setup procedures via ***Direct***SOFT.

CPU Modes

There are two possible operating modes available with DL105 Micro PLCs.

- RUN — executes program and updates I/O points.
- PROGRAM — allows program changes. The CPU halts execution of the ladder program and all output points are turned off.

Mode of Operation at Power-up

The DL105 operates as follows when the power is connected.

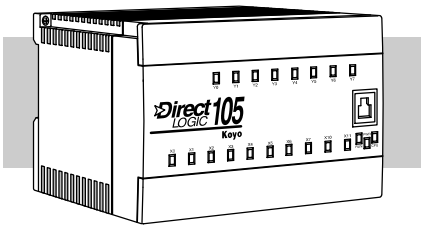
1. The DL105 CPU will normally power-up in the mode that it was in just prior to the power interruption. For example, if the CPU was in Program Mode when the power was disconnected, the CPU will power-up in Program Mode (see warning note below).
2. You can configure the DL105 to always power-up in the Run Mode. You can set bit 13 in V7633 (nonvolatile memory) to enable this feature; we'll show you how to set the bit later in this chapter.



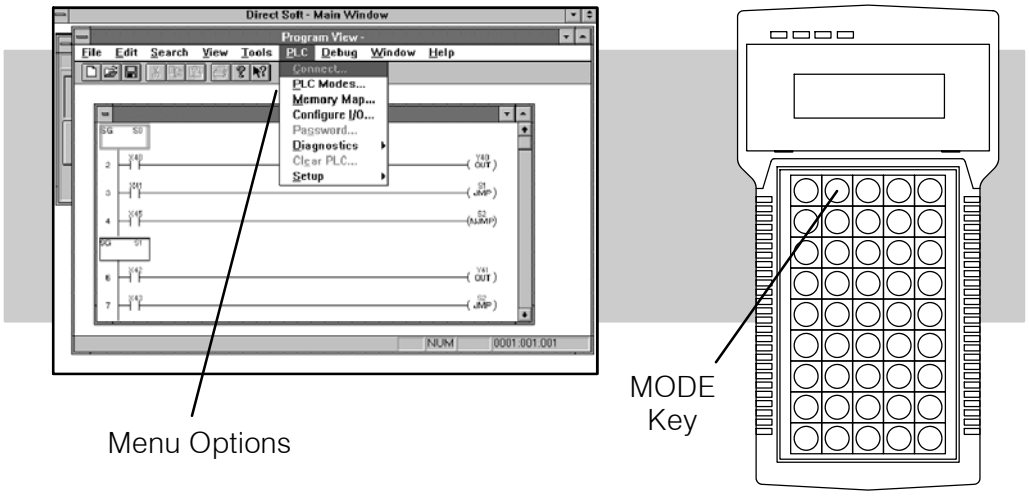
WARNING: If bit 13 in memory location V7633 is *not set*, once the super capacitor has discharged the system memory may not retain the previous mode of operation. When this occurs, the PLC can power-up in either Run or Program Mode. There is no way to determine which mode will be entered. Failure to adhere to this warning greatly increases the risk of unexpected equipment startup.

Changing Modes in the DL105 PLC

The DL105 Micro PLC does not have an external switch to switch CPU operating modes. You have to use a programming device, such as the handheld programmer or ***Direct***SOFT, to change the operating mode.



You can use either ***Direct***SOFT or the Handheld Programmer to change the CPU mode of operation. With ***Direct***SOFT you use a menu option in the PLC menu. With the Handheld Programmer, you use the MODE key.

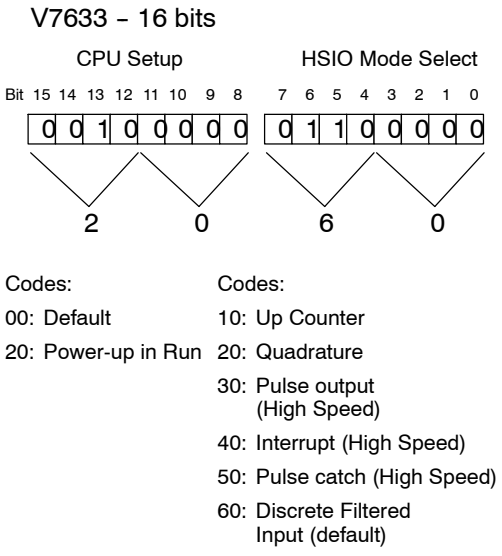


Setting Bits in V7633

You can use the Handheld Programmer or ***Direct***SOFT to set the proper bits in V7633.

Since you cannot access the bits individually, you have to enter a constant that will result in the appropriate bit being set. The first two digits of the constant are used to select the CPU options. The second two digits are used with the High-Speed I/O function to select various options. If you're using High Speed I/O functions, make sure you also enter the appropriate code for the feature selected If you want the HSIO inputs and outputs to default to regular I/O point operation, just enter 60 as the last two digits of the code. This configures all I/O points to operate only as standard discrete I/O.

The diagram shows how the upper and lower bytes of V7633 are used. For example, if you entered 2060 into V7633, the powerup-in-run option is selected, and the discrete filtered inputs are selected.



The following keystrokes show how you can enter the codes into V7633 with the D2-HPP Handheld Programmer.

Select V7633 for Monitoring

CLR CLR SHFT

V AND | H 7 | G 6 | D 3 | D 3 | STAT

```
V 7634  V7633
  0001   0000
```

Enter the Code

SHFT | K JMP | C 2 | A 0 | G 6 | A 0 | ENT

```
 V 7634  V7633
K2060
```

Since the changes take affect immediately, you may receive an error message. For example, if you select Power-up in Run Mode and the CPU does not yet contain a program, an error will occur.

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 ***Direct***SOFT 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 ***Direct***SOFT package. The following table shows a list of the Auxiliary functions for the Handheld Programmer.

AUX 2* — RLL Operations	
21	Check Program
22	Change Reference
23	Clear Ladder Range
24	Clear All Ladders
AUX 3* — V-Memory Operations	
31	Clear V-Memory
AUX 4* — I/O Configuration	
41	Show I/O Configuration
AUX 5* — CPU Configuration	
51	Modify Program Name
53	Display Scan Time
54	Initialize Scratchpad
55	Set Watchdog Timer
57	Set Retentive Ranges
58	Test Operations
5B	HSIO Configuration

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

Clearing an Existing Program

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

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

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

Initializing System Memory

The DL105 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.

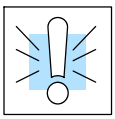
Remember, this AUX function will reset all system memory. If you have set special parameters such as retentive ranges, etc. they will be erased when AUX 54 is used. Make sure you that you have considered all ramifications of this operation before you select it.

Setting Retentive Memory Ranges

The DL105 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. (see Appendix E) The default settings are:

Memory Area	DL105	
	Default Range	Available Range
Control Relays	C300 – C377	C0 – C377
V-Memory	V2000 – V2377	V0 – V2377
Timers	None by default	T0 – T77
Counters	CT0 – CT77	CT0 – CT77
Stages	None by default	S0 – S377

You can use AUX 57 (see Appendix A) to set the retentive ranges. You can also use ***Direct***SOFT menus to select the retentive ranges.

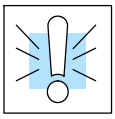


WARNING: The DL105 PLCs do not have battery back-up. The super capacitor will retain the values in the event of a power loss, but only for a short period of time, depending on conditions. If the retentive ranges are important for your application, make sure you program critical parameters into EEPROM locations.

Using a Password

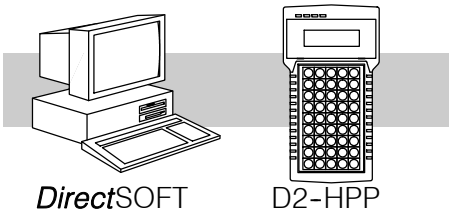
The DL105 PLCs allow you to use a password to help minimize the risk of unauthorized program and/or data changes. Once you enter a password you can "lock" the PLC against access. Once the CPU is locked you must enter the password before you can use a programming device to change any system parameters.

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 removed.

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



Select AUX 81

CLR | CLR | I 8 | B 1 | AUX | ENT

```
PASSWORD
00000000
```

Enter the new 8-digit password

X | X | • • • | X | ENT

```
PASSWORD
XXXXXXXX
```

Press CLR to clear the display

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

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

CPU Operation

Achieving the proper control for your equipment or process requires a good understanding of how DL105 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 — DL105 CPUs offer a wide variety of resources, such as timers, counters, inputs, etc. The memory map section shows the organization and availability of these data types.

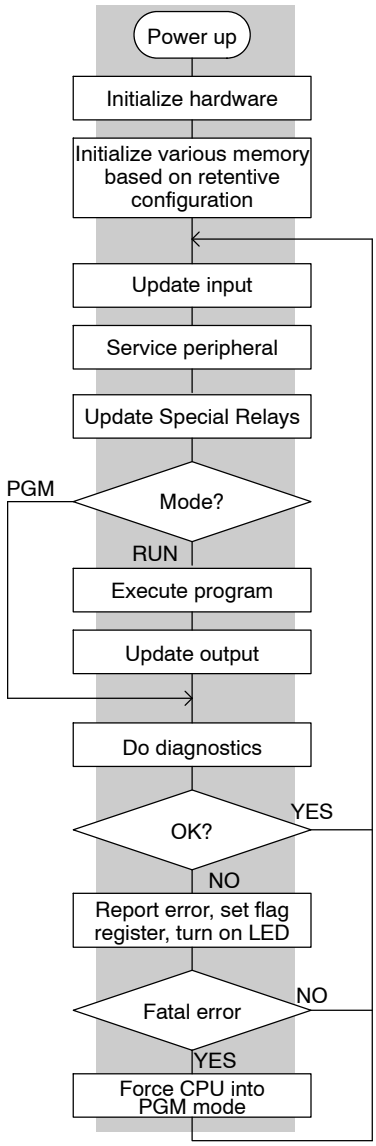
CPU Operating System

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

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

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

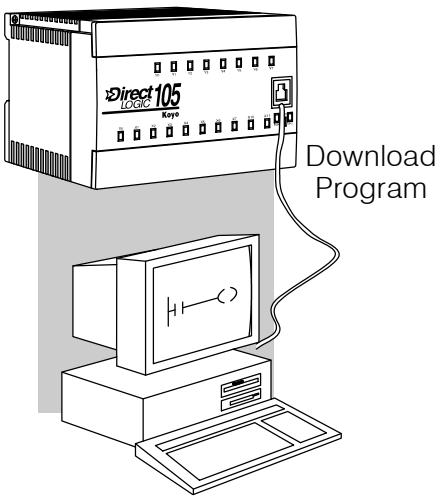
Error detection has two levels. Non-fatal errors are reported, but the CPU remains in its current mode. If a fatal error occurs, the CPU is forced into program mode and the outputs go off.



Program Mode

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

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



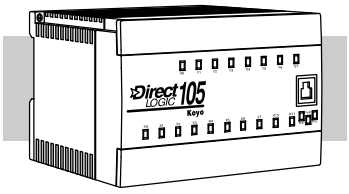
Run Mode

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:

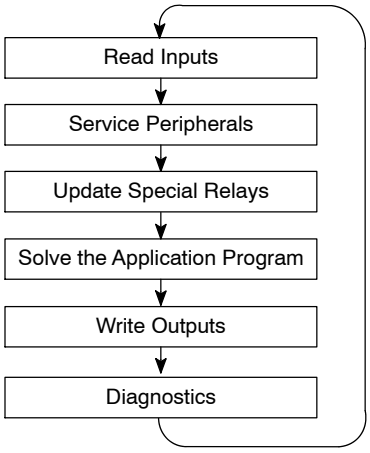
- Monitor and change I/O point status
- Update timer/counter preset values
- Update Variable memory locations

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.

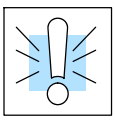
You can use ***Direct***SOFT or the D2-HPP Handheld Programmer to place the CPU in Run Mode.



Normal Run mode scan



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 8.



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

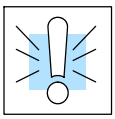
After the CPU reads the inputs from the input modules, it reads any attached peripheral devices. This is primarily a communications service for any attached devices. For example, it would read a programming device to see if any input, output, or other memory type status needs to be modified.

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

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

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

The DL105 PLCs only retain the forced value for one scan. There is an exception to this rule. For example, if the point address is greater than X11 or Y7 or it is not used in the ladder program, then the point will maintain the forced status.



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

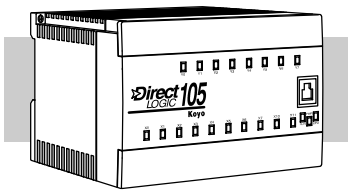
Update Special Relays and Special Registers

There are certain V-memory locations that contain 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.

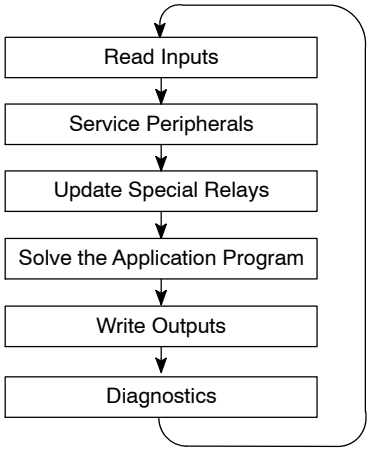
Solve Application Program

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.



Normal Run mode scan



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.

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 DL105 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 DL105 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 to determine the amount of time spent during the various segments of operation.

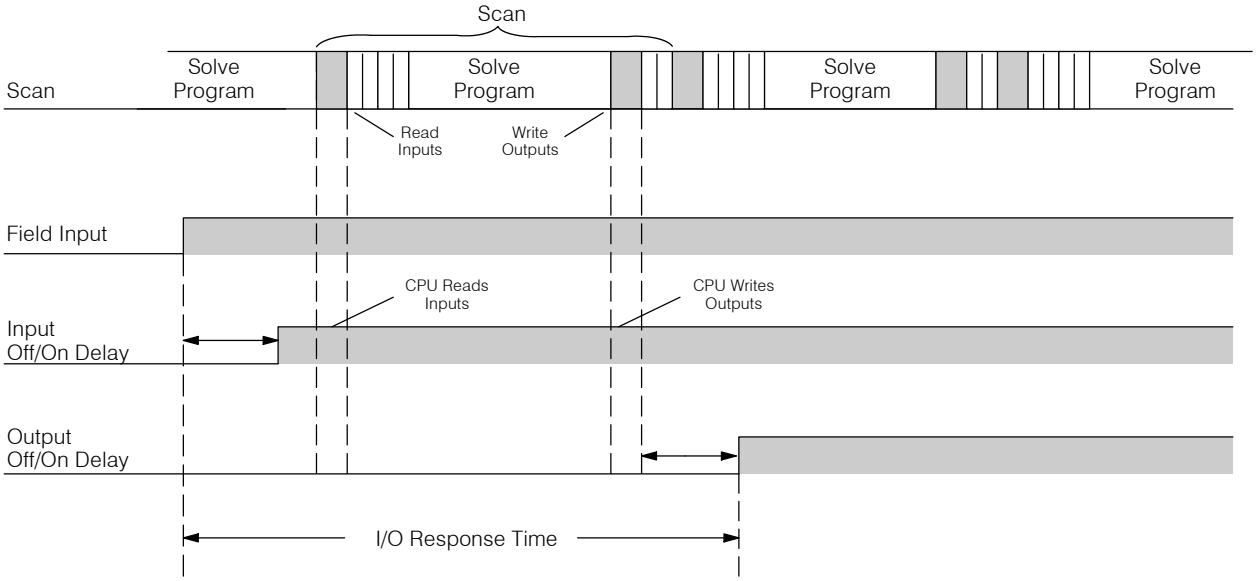
There are four things that can affect the I/O response time.

- 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.

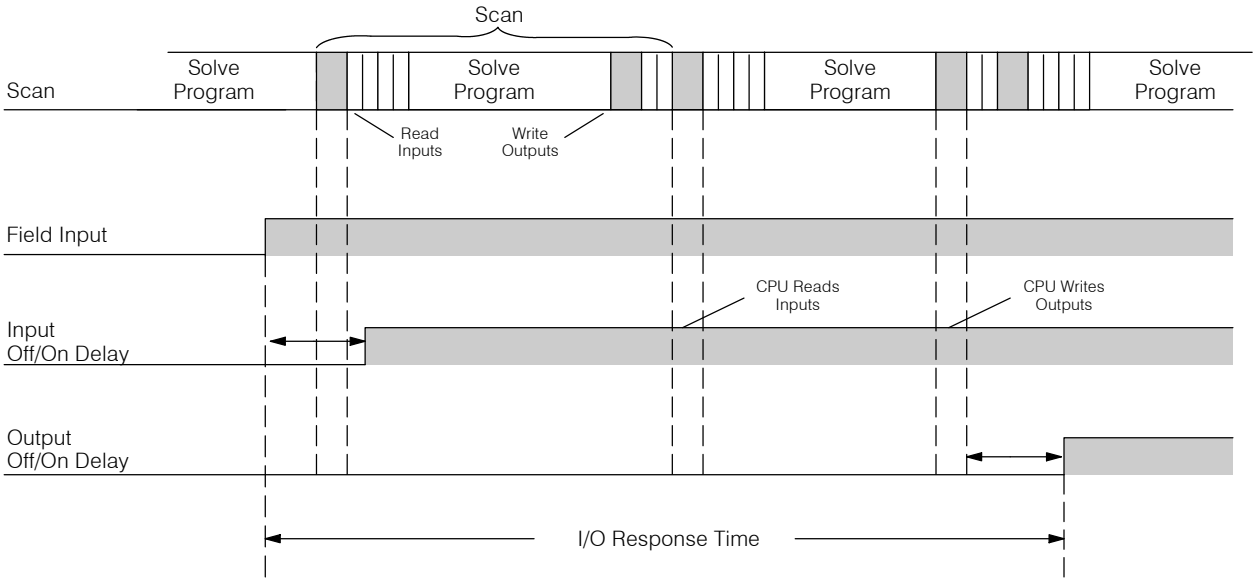


In this case, you can calculate the response time by simply adding the following items:

Input Delay + Scan Time + Output Delay = 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 does not get read until the following scan. The following diagram shows an example of the timing for this situation.



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

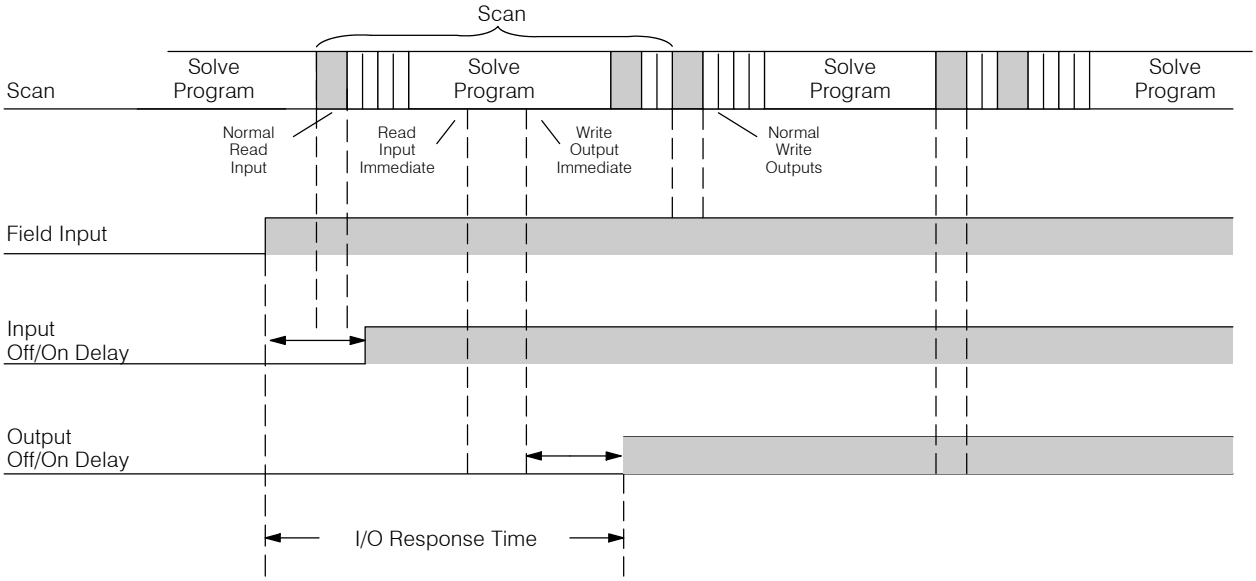
Input Delay +(2 x Scan Time) + Output Delay = Response Time

Improving Response Time

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

- 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 the Chapter 3 for details on using this feature.

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



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

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

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



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

CPU Scan Time Considerations

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

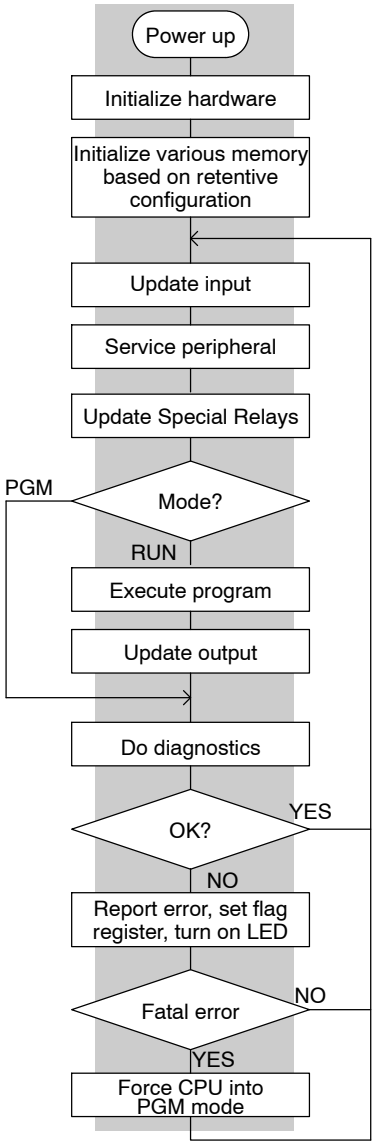
As we've shown previously there are several segments that make up the scan cycle. Each of these segments requires a certain amount of time to complete. Of all the segments, the following are the most important.

- 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 μS. 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 μS. Don't confuse this with the I/O response time that was discussed earlier.

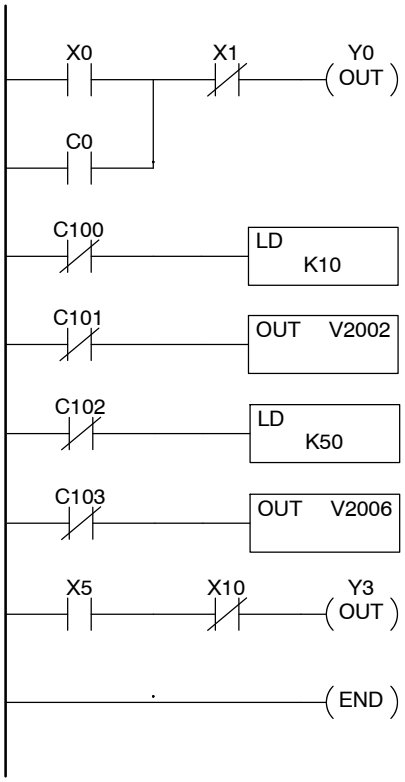
Application Program Execution

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 DL105 Micro PLC. For example, the execution time for running the program shown below is calculated as follows:

Instruction	Time
STR X0	3.3 µs
OR C0	2.7 µs
ANDN X1	2.7 µs
OUT Y0	3.4 µs
STRN C100	3.9 µs
LD K10	62 µs
STRN C101	3.9 µs
OUT V2002	60 µs
STRN C102	3.9 µs
LD K50	62 µs
STRN C103	3.9 µs
OUT V2006	60 µs
STR X5	3.3 µs
ANDN X10	2.7 µs
OUT Y3	3.4 µs
END	27 µs
TOTAL	308.1 µs

Overhead	DL105
Minimum	0.86 mS
Maximum	3.85 ms



The program above takes only 308.1 µs to execute during each scan. The total scan time is the sum of the program execution plus the overhead as shown above. "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.

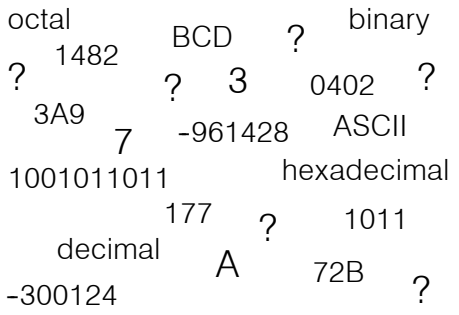


NOTE: You can move words to EEPROM from within the application program. This can add up to 10ms per 32 word boundary.

Program Control Instructions — the DL105 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 mS period interrupts the main program execution (before the END statement) every 10 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 ***Direct***LOGIC PLCs for the first time, please take a moment to study how our PLCs use numbers. You'll find that each PLC manufacturer has 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 ***Direct***LOGIC PLCs. The information you learn here applies to all our PLCs!

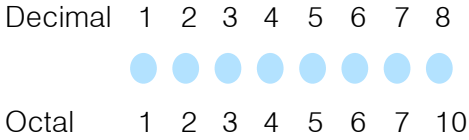


As any good computer does, PLCs store and manipulate numbers in binary form: just ones and zeros. So why do we have to deal with numbers in so many different forms? Numbers have meaning, and some *representations* are more convenient than others for particular purposes. Sometimes we use numbers to represent 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 G for numbering system details).

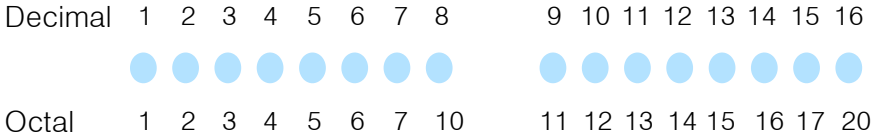
PLC Resources

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

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

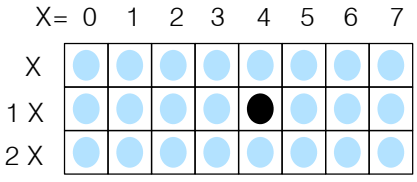


In the figure below, we have two groups of eight circles. Counting in octal we have "20" items, meaning 2 groups of eight, plus 0 individuals Don't say "twenty", say "two-zero octal". This makes a clear distinction between number systems.



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

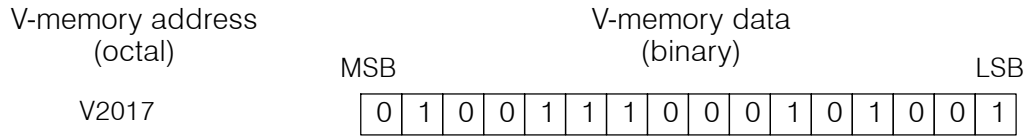
Our circles are in an array of square containers to the right. To access a resource, our PLC instruction will address its location using the octal references shown. If these were counters, "CT14" would access the black circle location.



V-Memory

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

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

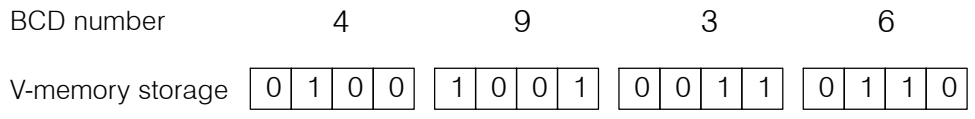


V-memory data is 16-bit binary, but we rarely program the data registers one bit at a time. We use instructions or viewing tools that let us work with decimal, octal, and hexadecimal numbers. All these are converted and stored as binary for us.

A frequently-asked question is "How do I tell if a number is octal, BCD, or hex"? The answer is that we usually cannot tell just by looking at the data... but it does not really matter. What matters is: the source or mechanism which writes data into a V-memory location and the thing which later reads it must both use the same data type (i.e., octal, hex, binary, or whatever). The V-memory location is just a storage box... that's all. It does not convert or move the data on its own.

Binary-Coded Decimal Numbers

Since humans naturally count in decimal (10 fingers, 10 toes), we prefer to enter and view PLC data in decimal as well. However, computers are more efficient in using pure binary numbers. A compromise solution between the two is Binary-Coded Decimal (BCD) representation. A BCD digit ranges from 0 to 9, and is stored as four binary bits (a nibble). This permits each V-memory location to store four BCD digits, with a range of decimal numbers from 0000 to 9999.



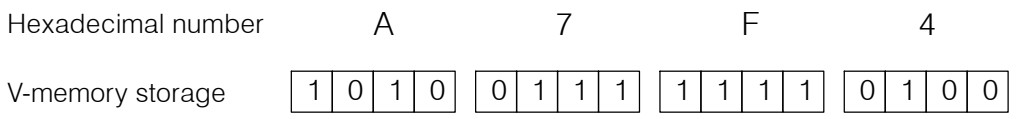
In a pure binary sense, a 16-bit word can represent numbers from 0 to 65535. In storing BCD numbers, the range is reduced to only 0 to 9999. Many math instructions use Binary-Coded Decimal (BCD) data, and ***Direct***SOFT and the handheld programmer allow us to enter and view data in BCD.

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.

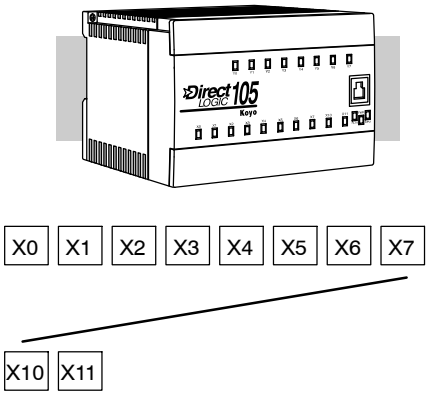


Memory Map

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 DL105 Micro PLCs. A memory map overview for the CPU follows the memory descriptions.

Octal Numbering System

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

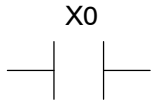


Discrete and Word Locations

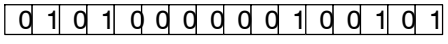
As you examine the different memory types, you'll notice two types of memory in the DL105, discrete and word memory. Discrete memory is one bit that can be either a 1 or a 0. Word memory is referred to as V-memory (variable) and is a 16-bit location normally used to manipulate data/numbers, store data/numbers, etc.

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

Discrete – On or Off, 1 bit



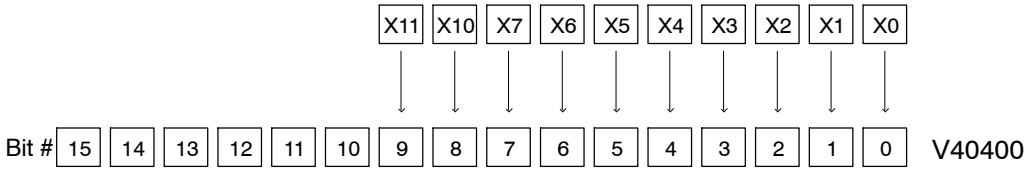
Word Locations – 16 bits



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.

10 Discrete (X) Input Points



These discrete memory areas and their corresponding V-memory ranges are listed in the memory area table for DL105 Micro PLCs on the following pages.

Input Points (X Data Type)

The discrete input points are noted by an X data type. There are 10 discrete input points available with DL105 CPUs. In this example, the output point Y0 will be turned on when input X0 energizes.



Output Points (Y Data Type)

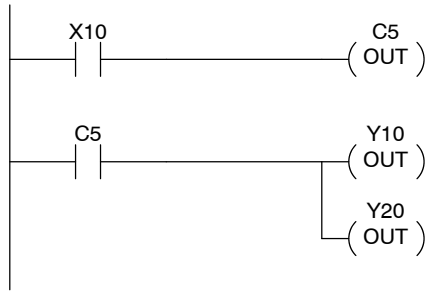
The discrete output points are noted by a Y data type. There are 128 discrete output points available with DL105 CPUs. In this example, output point Y1 will be turned on when input X1 energizes.



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.

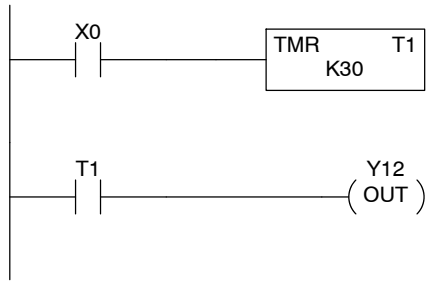
In this example, memory location C5 will energize when input X10 turns on. The second rung shows a simple example of how to use a control relay as an input.



Timers and Timer Status Bits (T Data type)

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

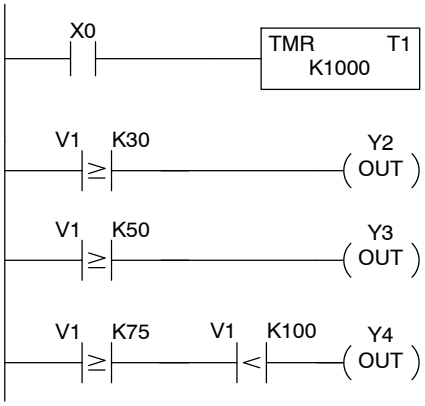
When input X0 turns on, timer T1 will start. When the timer reaches the preset of 3 seconds (K of 30) timer status contact T1 turns on. When T1 turns on, output Y12 turns on. Turning off X0 resets the timer.



Timer Current Values (V Data Type)

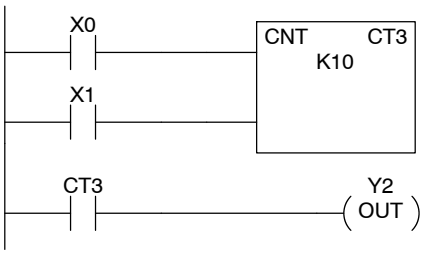
As mentioned earlier, some information is automatically stored in V-memory. This is true for the current values associated with timers. For example, V0 holds the current value for Timer 0, V1 holds the current value for Timer 1, etc.

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



Counters and Counter Status Bits (CT Data type)

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

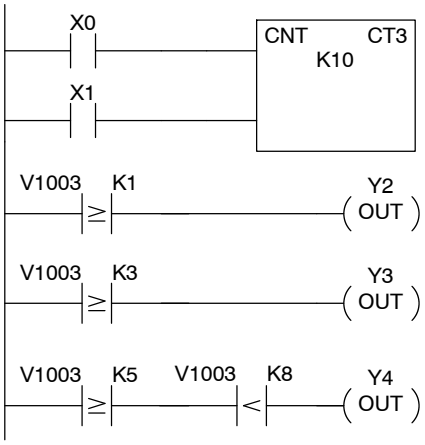


Each time contact X0 transitions from off to on, the counter increments by one. (If X1 comes on, the counter is reset to zero.) When the counter reaches the preset of 10 counts (K of 10) counter status contact CT3 turns on. When CT3 turns on, output Y12 turns on.

Counter Current Values (V Data Type)

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

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

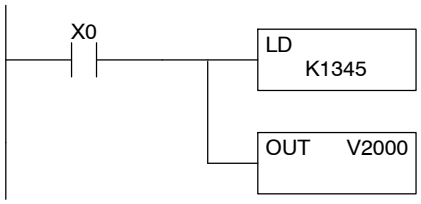


Word Memory (V Data Type)

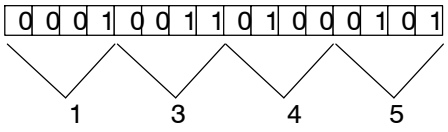
Word memory is referred to as V-memory (variable) and is a 16-bit location normally used to manipulate data/numbers, store data/numbers, etc. (see Appendix E).

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

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



Word Locations – 16 bits

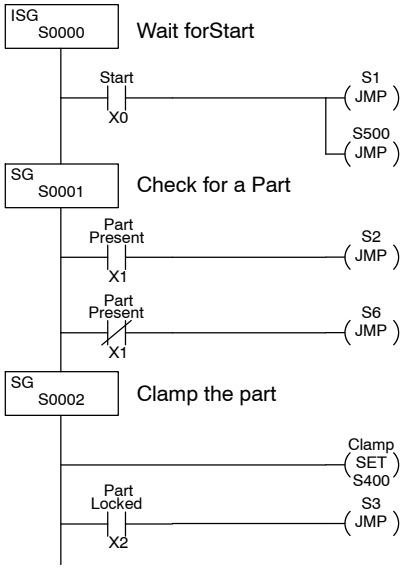


Stages (S Data type)

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

DL105 System V-memory

System Parameters and Default Data Locations (V Data Type)

The DL105 PLCs reserve several V-memory locations for storing system parameters or certain types of system data. These memory locations store things like the error codes, High-Speed I/O data, and other types of system setup information.

System V-memory	Description of Contents	Default Values / Ranges
V2320–V2377	The default location for multiple preset values for the High-Speed Counter	N/A
V7620–V7627	Locations for DV–1000 operator interface parameters	
V7620	Sets the V-memory location that contains the value.	V0 – V2377
V7621	Sets the V-memory location that contains the message.	V0 – V2377
V7622	Sets the total number (1 – 16) of V-memory locations to be displayed.	1 – 16
V7623	Sets the V-memory location that contains the numbers to be displayed.	V0 – V2377
V7624	Sets the V-memory location that contains the character code to be displayed.	V0 – V2377
V7625	Contains the function number that can be assigned to each key.	V-memory location for X, Y, or C points used.
V7626	Powerup operational mode.	0, 1, 2, 12, 3
V7627	Change preset value.	0000 to 9999
V7630	Starting location for the multi–step presets for channel 1. The default value is 2320, which indicates the first value should be obtained from V2320. Since there are 24 presets available, the default range is V2320 – V2377. You can change the starting point if necessary.	Default: V2320 Range: V0 – V2320
V7631–V7632	Not used	N/A
V7633	Sets the desired function code for the high speed counter, interrupt, pulse catch, pulse train, and input filter. Location is also used for setting 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
V7634	X0 Setup Register for High-Speed I/O functions	Default: 1006
V7635	X1 Setup Register for High-Speed I/O functions	Default: 1006
V7636	X2 Setup Register for High-Speed I/O functions	Default: 1006
V7637	X3 Setup Register for High-Speed I/O functions	Default: 1006

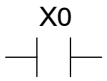
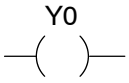
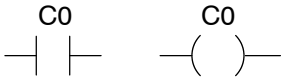
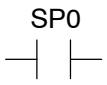
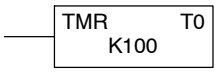
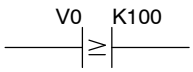
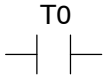
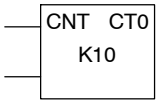
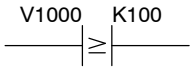
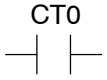
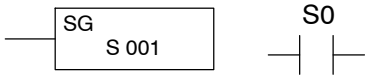
System V-memory	Description of Contents	Default Values / Ranges
V7640-V7642	Locations for DV-1000 operator interface parameters.	
V7640	Title Timer preset value pointer.	V2000-V2377
V7641	Titled Counter preset value pointer.	V2000-V2377
V7642	HiByte-Titled Timer preset block size, LoByte-Titled counter preset block size.	1-99
V7643-V7647	Not used	
V7751	Fault Message Error Code — stores the 4-digit code used with the FAULT instruction when the instruction is executed.	N/A
V7752-V7754	Not used	N/A
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	Not used	
V7763	Program address where syntax error exists	N/A
V7764	Syntax error code	N/A
V7765	Scan — stores the total number of scan cycles that have occurred since the last Program Mode to Run Mode transition.	N/A
V7666-V7774	Not used	N/A
V7775	Scan — stores the current scan time (milliseconds).	N/A
V7776	Scan — stores the minimum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds).	N/A
V7777	Scan — stores the maximum scan time that has occurred since the last Program Mode to Run Mode transition (milliseconds).	N/A

DL105 Aliases

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

Address Start	Alias Start	Example
V0	TA0	V0 is the timer accumulator value for timer 0, therefore, it's alias is TA0. TA1 is the alias for V1, etc..
V1000	CTA0	V1000 is the counter accumulator value for counter 0, therefore, it's alias is CTA0. CTA1 is the alias for V1001, etc.
V40000	VGX0	V40000 is the word memory reference for discrete bits GX0 through GX17, therefore, it's alias is VGX0. V40001 is the word memory reference for discrete bits GX20 through GX 37, therefore, it's alias is VGX20.
V40200	VGY0	V40200 is the word memory reference for discrete bits GY0 through GY17, therefore, it's alias is VGY0. V40201 is the word memory reference for discrete bits GY20 through GY 37, therefore, it's alias is VGY20.
V40400	VX0	V40400 is the word memory reference for discrete bits X0 through X17, therefore, it's alias is VX0. V40401 is the word memory reference for discrete bits X20 through X37, therefore, it's alias is VX20.
V40500	VY0	V40500 is the word memory reference for discrete bits Y0 through Y17, therefore, it's alias is VY0. V40501 is the word memory reference for discrete bits Y20 through Y37, therefore, it's alias is VY20.
V40600	VC0	V40600 is the word memory reference for discrete bits C0 through C17, therefore, it's alias is VC0. V40601 is the word memory reference for discrete bits C20 through C37, therefore, it's alias is VC20.
V41000	VS0	V41000 is the word memory reference for discrete bits S0 through S17, therefore, it's alias is VS0. V41001 is the word memory reference for discrete bits S20 through S37, therefore, it's alias is VS20.
V41100	VT0	V41100 is the word memory reference for discrete bits T0 through T17, therefore, it's alias is VT0. V41101 is the word memory reference for discrete bits T20 through T37, therefore, it's alias is VT20.
V41140	VCT0	V41140 is the word memory reference for discrete bits CT0 through CT17, therefore, it's alias is VCT0. V41141 is the word memory reference for discrete bits CT20 through CT37, therefore, it's alias is VCT20.
V41200	VSP0	V41200 is the word memory reference for discrete bits SP0 through SP17, therefore, it's alias is VSP0. V41201 is the word memory reference for discrete bits SP20 through SP37, therefore, it's alias is VSP20.

DL105 Memory Map

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

1 – The DL105 systems are limited to 10 discrete inputs and 8 discrete outputs with the present available hardware, but 128 point addresses exist.

X Input Bit Map

This table provides a listing of individual Input points associated with each V-memory address bit for the DL105's ten physical inputs. Actual available references are X0 to X177 (V40400 – V40407).

MSB						DL105 Input (X) Points									LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
–	–	–	–	–	–	011	010	007	006	005	004	003	002	001	000	V40400

Y Output Bit Map

This table provides a listing of individual output points associated with each V-memory address bit for the DL105's eight physical outputs. Actual available references are Y0 to Y177 (V40500 – V40507).

MSB						DL105 Output (Y) Points									LSB	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	V40500

Control Relay Bit Map

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

MSB						DL105 Control Relays (C)									LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000	V40600
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020	V40601
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040	V40602
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060	V40603
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100	V40604
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120	V40605
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140	V40606
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160	V40607
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200	V40610
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220	V40611
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240	V40612
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260	V40613
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300	V40614
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320	V40615
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340	V40616
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360	V40617

Stage Control / Status Bit Map

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

MSB					DL105 Stage (S) Control Bits											LSB	Address
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
017	016	015	014	013	012	011	010	007	006	005	004	003	002	001	000		V41000
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020		V41001
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040		V41002
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060		V41003
117	116	115	114	113	112	111	110	107	106	105	104	103	102	101	100		V41004
137	136	135	134	133	132	131	130	127	126	125	124	123	122	121	120		V41005
157	156	155	154	153	152	151	150	147	146	145	144	143	142	141	140		V41006
177	176	175	174	173	172	171	170	167	166	165	164	163	162	161	160		V41007
217	216	215	214	213	212	211	210	207	206	205	204	203	202	201	200		V41010
237	236	235	234	233	232	231	230	227	226	225	224	223	222	221	220		V41011
257	256	255	254	253	252	251	250	247	246	245	244	243	242	241	240		V41012
277	276	275	274	273	272	271	270	267	266	265	264	263	262	261	260		V41013
317	316	315	314	313	312	311	310	307	306	305	304	303	302	301	300		V41014
337	336	335	334	333	332	331	330	327	326	325	324	323	322	321	320		V41015
357	356	355	354	353	352	351	350	347	346	345	344	343	342	341	340		V41016
377	376	375	374	373	372	371	370	367	366	365	364	363	362	361	360		V41017

Timer Status Bit Map

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

MSB					DL105 Timer (T) 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		V41100
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020		V41101
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040		V41102
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060		V41103

Counter Status Bit Map

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

MSB					DL105 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		V41140
037	036	035	034	033	032	031	030	027	026	025	024	023	022	021	020		V41141
057	056	055	054	053	052	051	050	047	046	045	044	043	042	041	040		V41142
077	076	075	074	073	072	071	070	067	066	065	064	063	062	061	060		V41143

5

Standard RLL Instructions

In This Chapter. . . .

- Boolean Instructions
- Comparative Boolean
- Immediate Instructions
- Timer, Counter and Shift Register Instructions
- Accumulator / Stack Load and Output Data Instructions
- Logical Instructions (Accumulator)
- Math Instructions
- Bit Operation Instructions (Accumulator)
- Number Conversion Instructions (Accumulator)
- Table Instructions
- CPU Control Instructions
- Program Control Instructions
- Interrupt Instructions
- Message Instructions

Introduction

DL105 Micro PLCs offer a wide variety of instructions to perform many different types of operations. This chapter shows you how to use each standard Relay Ladder Logic (RLL) instruction. In addition to these instructions, you may also need to refer to the Drum instruction in Chapter 6, or the Stage programming instructions in Chapter 7.

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

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

Instruction	Page
ACON	5-84
ADD	5-60
ADDD	5-61
AND	5-11, 5-22, 5-52
AND STR	5-12
ANDD	5-53
ANDE	5-19
ANDI	5-24
ANDN	5-11, 5-22
ANDNE	5-19
ANDNI	5-24
BCD	5-72
BIN	5-71
CMP	5-58
CMPD	5-59
CNT	5-33
DECB	5-66
DECO	5-70
DISI	5-80
DIV	5-65
DLBL	5-84
EDRUM	6-2, 6-12
ENCO	5-69
END	5-77
ENI	5-80
FAULT	5-83
INCB	5-66
INT	5-80

Instruction	Page
INV	5-73
IRT	5-80
ISG	7-20
JMP	7-20
LD	5-45
LDA	5-48
LDD	5-46
LDF	5-47
LDLBL	5-75
MLR	5-78
MLS	5-78
MOV	5-74
MOVMC	5-75
MUL	5-64
NCON	5-84
NOP	5-77
OR	5-10, 5-21, 5-54
OR OUT	5-14
OR OUTI	5-25
OR STR	5-12
ORD	5-55
ORE	5-18
ORI	5-23
ORN	5-10, 5-21
ORNE	5-18
ORNI	5-23
OUT	5-14, 5-49
OUTD	5-49
OUTF	5-50

Instruction	Page
PAUSE	5-16
PD	5-15
POP	5-50
RST	5-15
RSTI	5-26
SET	5-15
SETI	5-26
SG	7-19
SGCNT	5-35
SHFL	5-67
SHFR	5-68
SR	5-39
STOP	5-77
STR	5-9, 5-20
STRE	5-17
STRI	5-23
STRN	5-9, 5-20
STRNE	5-17
STRNI	5-23
SUB	5-62
SUBD	5-63
TMR	5-28
TMRF	5-28
TMRA	5-30
TMRAF	5-30
UDC	5-37
XOR	5-56
XORD	5-57

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 ***Direct***SOFT software is a similar program. It allows you to use graphic symbols to develop the program; therefore, you don't necessarily have to know the instruction mnemonics in order to develop a program.

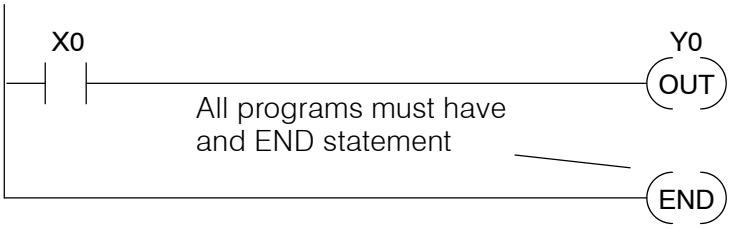
Many of the instructions in this chapter are not program instructions used in ***Direct***SOFT, but are implied. In other words, they are not actually keyboard commands, however, they can be seen in a Mnemonic View of the program once the ***Direct***SOFT program has been developed and accepted (compiled).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

DS	Implied
HPP	Used

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

END Statement

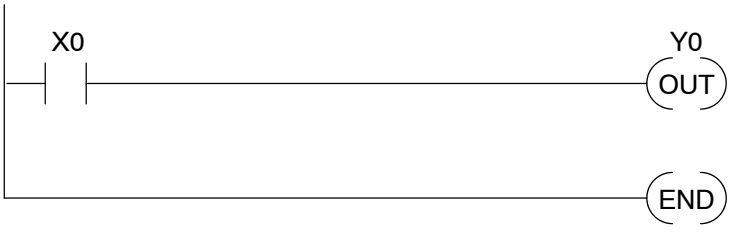
All DL105 programs require an END statement as the last instruction. This tells the CPU that this is the end of the program. Normally, any instructions placed after the END statement will not be executed. There are exceptions to this such as interrupt routines, etc. This chapter will discuss the instruction 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.

DirectSOFT Example



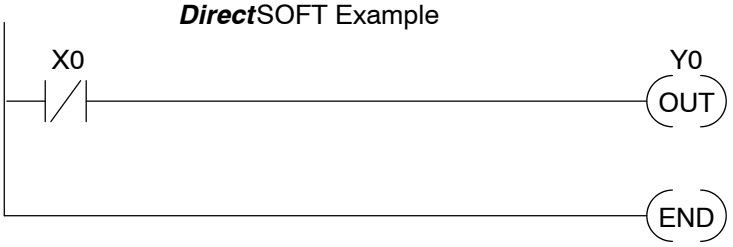
Handheld Mnemonics

```
STR X0
OUT Y0
END
```

Normally Closed Contact

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

DirectSOFT Example



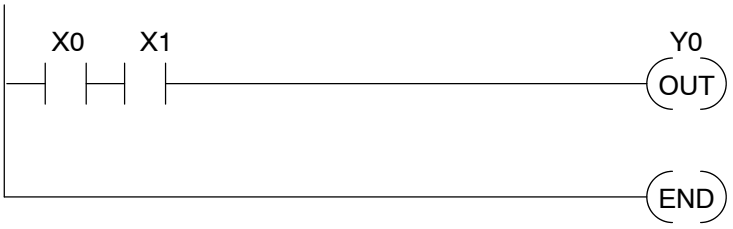
Handheld Mnemonics

```
STRN X0
OUT Y0
END
```

Contacts in Series

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

DirectSOFT Example



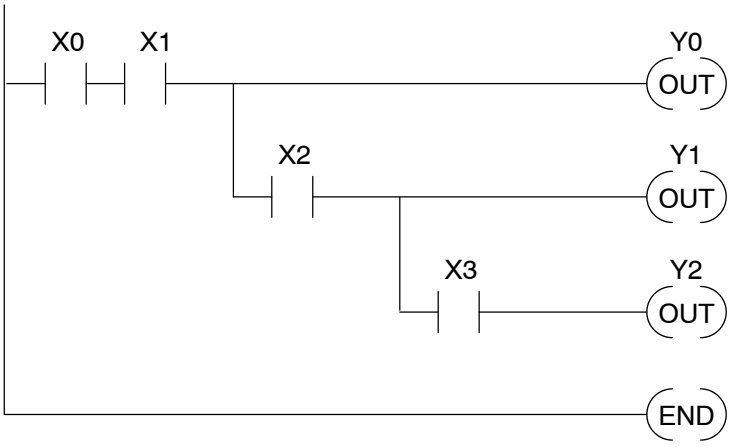
Handheld Mnemonics

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

Midline Outputs

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

DirectSOFT Example



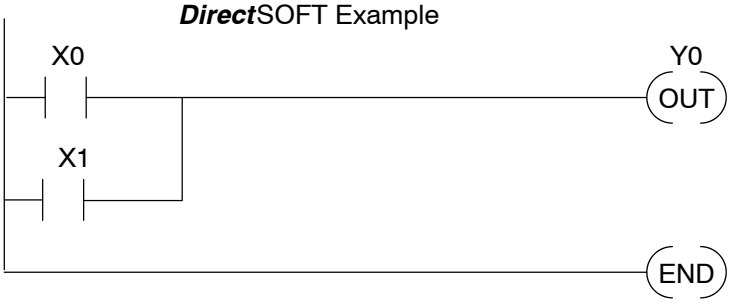
Handheld Mnemonics

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

Parallel Elements

You also have to join contacts in parallel. The OR instruction allows you to do this. The following example shows two contacts in parallel and a single output coil. The instructions would be STR X0, OR X1, followed by OUT Y0.

DirectSOFT Example



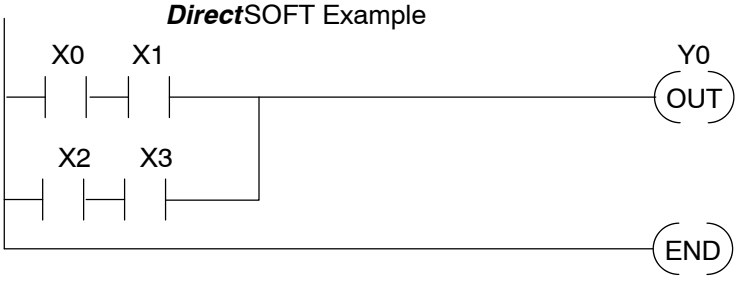
Handheld Mnemonics

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

Joining Series Branches in Parallel

Quite often it is necessary to join several groups of series elements in parallel. The Or Store (ORSTR) instruction allows this operation. The following example shows a simple network consisting of series elements joined in parallel.

DirectSOFT Example



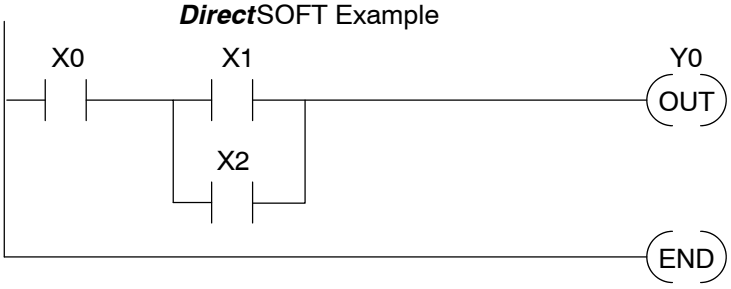
Handheld Mnemonics

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

Joining Parallel Branches in Series

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.

DirectSOFT Example

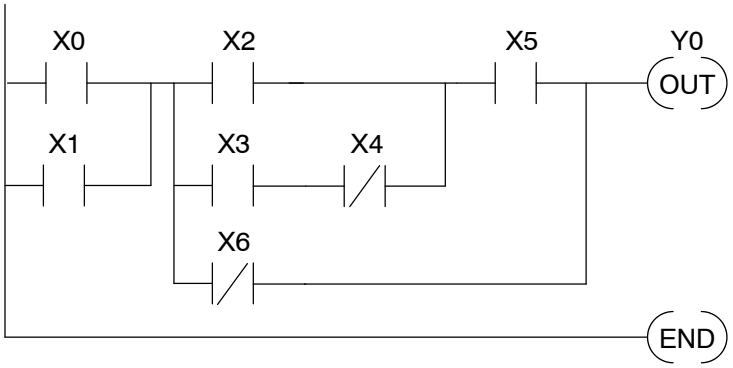


Handheld Mnemonics

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

Combination Networks

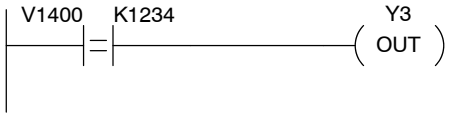
You can combine the various types of series and parallel branches to solve most any application problem. The following example shows a simple combination network.



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 DL105 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.

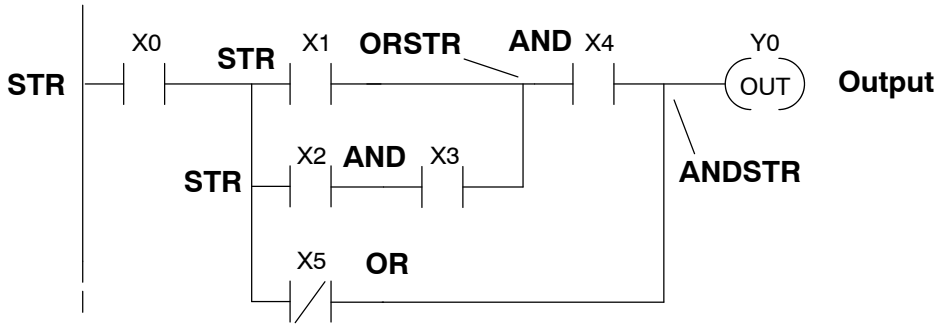
In the following example when the value in V-memory location V1400 is equal to the constant value 1234, Y3 will energize.



Boolean Stack

There are limits to how many elements you can include in a rung. This is because the DL105 PLCs use an 8-level boolean stack to evaluate the various logic elements. The boolean stack is a temporary storage area that solves the logic for the rung. Each time the program encounters a STR instruction, the instruction is placed on the top of the stack. Any other STR instructions already on the boolean stack are pushed down a level. The ANDSTR, and ORSTR instructions combine levels of the boolean stack when they are encountered. An error will occur during program compilation if the CPU encounters a rung that uses more than the eight levels of the boolean stack.

The following example shows how the boolean stack is used to solve boolean logic.



STR X0

1	STR X0
2	
3	
4	
5	
6	
7	
8	

STR X1

1	STR X1
2	STR X0
3	
4	
5	
6	
7	
8	

STR X2

1	STR X2
2	STR X1
3	STR X0
4	
5	
6	
7	
8	

AND X3

1	X2 AND X3
2	STR X1
3	STR X0
4	
5	
6	
7	
8	

ORSTR

1	X1 OR (X2 AND X3)
2	STR X0
3	
.	
.	
.	
8	

AND X4

1	X4 AND [X1 OR (X2 AND X3)]
2	STR X0
3	
.	
.	
.	
8	

ORNOT X5

1	NOT X5 OR X4 AND [X1 OR (X2 AND X3)]
2	STR X0
3	
.	
.	
.	
8	

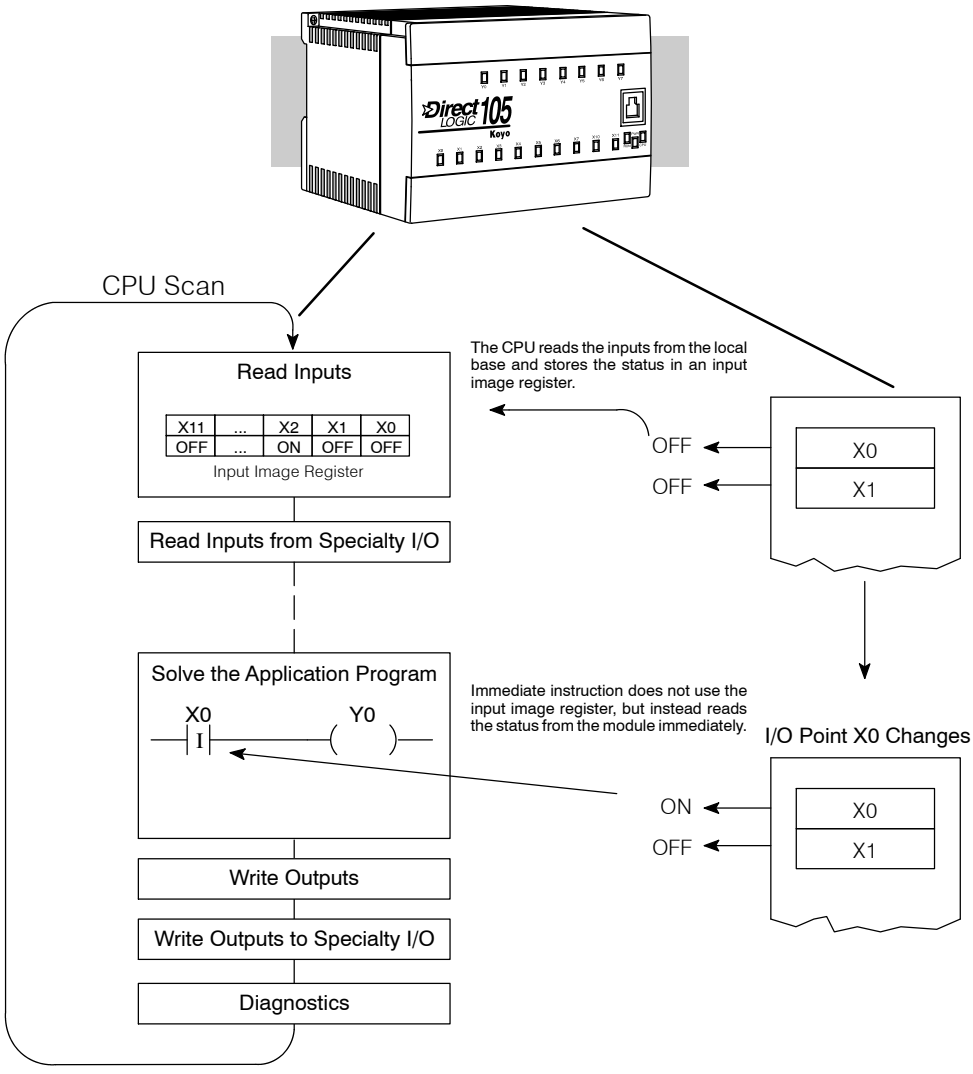
ANDSTR

1	X0 AND (NOT X5 OR X4) AND [X1 OR (X2 AND X3)]
2	
3	
.	
.	
.	
8	

Immediate Boolean The DL105 Micro PLCs usually can complete an operation cycle in a matter of milliseconds. However, in some applications you may not be able to wait a few milliseconds until the next I/O update occurs. The DL105 PLCs offer Immediate input and outputs which are special boolean instructions that allow reading directly from inputs and writing directly to outputs during the program execution portion of the CPU cycle. You may recall that this is normally done during the input or output update portion of the CPU cycle. The immediate instructions take longer to execute because the program execution is interrupted while the CPU reads or writes the I/O point. This function is not normally done until the read inputs or the write outputs portion of the CPU cycle.



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

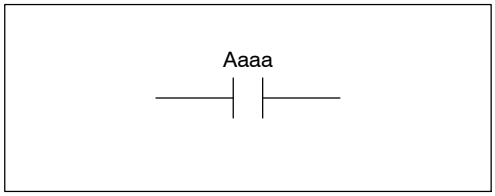


Boolean Instructions

Store (STR)

DS	Used
HPP	Used

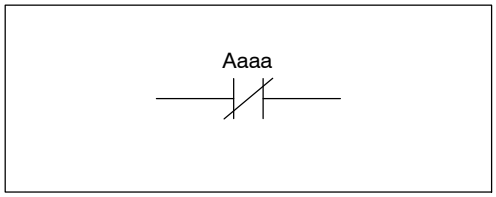
The Store instruction begins a new rung or an additional branch in a rung with a normally open contact. Status of the contact will be the same state as the associated image register point or memory location.



Store Not (STRN)

DS	Used
HPP	Used

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



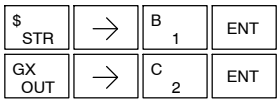
Operand Data Type		DL130 Range
	A	**aaa**
Inputs	X	0–11
Outputs	Y	0–7
Control Relays	C	0–377
Stage	S	0–377
Timer	T	0–77
Counter	CT	0–77
Special Relay	SP	0–117, 540–577

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

DirectSOFT



Handheld Programmer Keystrokes

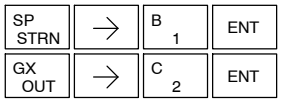


In the following Store Not example, when input X1 is off output Y2 will energize.

DirectSOFT



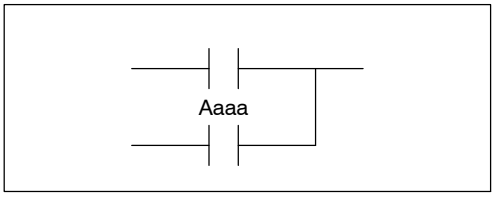
Handheld Programmer Keystrokes



Or (OR)

DS	Implied
HPP	Used

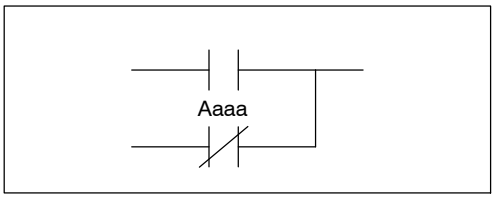
The Or instruction logically ors a normally open contact in parallel with another contact in a rung. The status of the contact will be the same state as the associated image register point or memory location.



Or Not (ORN)

DS	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		DL130 Range
	A	**aaa**
Inputs	X	0–11
Outputs	Y	0–7
Control Relays	C	0–377
Stage	S	0–377
Timer	T	0–77
Counter	CT	0–77
Special Relay	SP	0–117, 540–577

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

***Direct*SOFT**



Handheld Programmer Keystrokes

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

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

***Direct*SOFT**



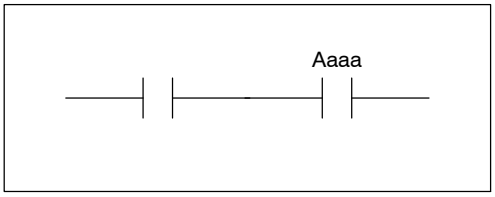
Handheld Programmer Keystrokes

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

And (AND)

DS	Implied
HPP	Used

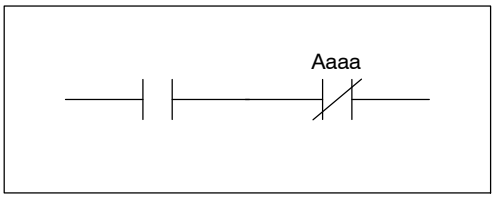
The And instruction logically ands a normally open contact in series with another contact in a rung. The status of the contact will be the same state as the associated image register point or memory location.



And Not (ANDN)

DS	Implied
HPP	Used

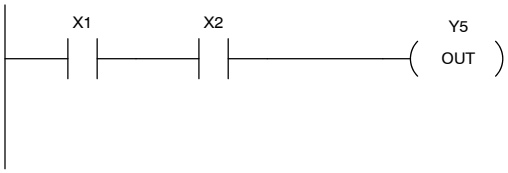
The And Not instruction logically ands a normally closed contact in series with another contact in a rung. The status of the contact will be opposite the state of the associated image register point or memory location.



Operand Data Type		DL130 Range
	A	**aaa**
Inputs	X	0–11
Outputs	Y	0–7
Control Relays	C	0–377
Stage	S	0–377
Timer	T	0–77
Counter	CT	0–77
Special Relay	SP	0–117, 540–577

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

DirectSOFT

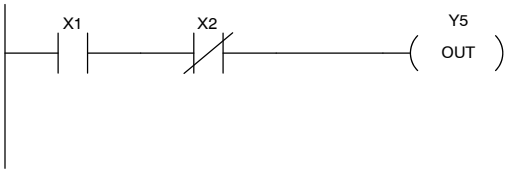


Handheld Programmer Keystrokes

$ 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



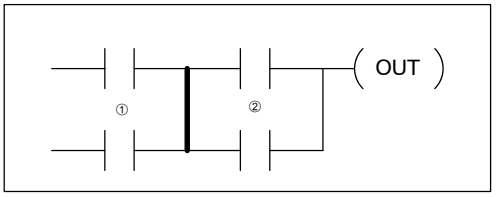
Handheld Programmer Keystrokes

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

And Store (AND STR)

DS	Implied
HPP	Used

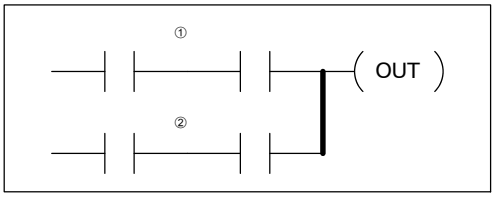
The And Store instruction logically ands two branches of a rung in series. Both branches must begin with the Store instruction.



Or Store (OR STR)

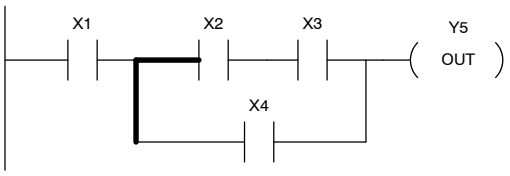
DS	Implied
HPP	Used

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.

***Direct*SOFT**

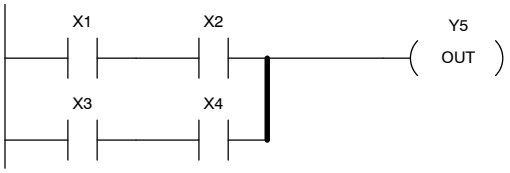


Handheld Programmer Keystrokes

$ STR	→	B 1	ENT
$ STR	→	C 2	ENT
V AND	→	D 3	ENT
Q OR	→	E 4	ENT
L ANDST	ENT		
GX OUT	→	F 5	ENT

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

***Direct*SOFT**

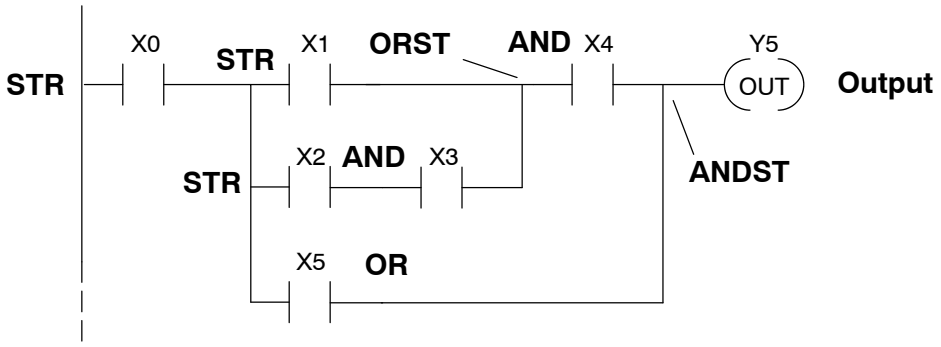


Handheld Programmer Keystrokes

$ STR	→	B 1	ENT
V AND	→	C 2	ENT
$ STR	→	D 3	ENT
V AND	→	E 4	ENT
M ORST	ENT		
GX OUT	→	F 5	ENT

There are limits to what you can enter with boolean instructions. This is because the DL105 internal CPU uses an 8-level stack to evaluate the various logic elements. The stack is a temporary storage area that helps solve the logic for the rung. Each time you enter a STR instruction, the instruction is placed on the top of the stack. Any other instructions on the stack are pushed down a level. The And Store and Or Store instructions combine levels of the stack when they are encountered. Since the stack is only eight levels, an error will occur if the CPU encounters a rung that uses more than the eight levels of the stack.

The following example shows how the stack is used to solve boolean logic.



STR X0

1	STR X0
2	
3	
4	
5	
6	
7	
8	

STR X1

1	STR X1
2	STR X0
3	
4	
5	
6	
7	
8	

STR X2

1	STR X2
2	STR X1
3	STR X0
4	
5	
6	
7	
8	

AND X3

1	X2 AND X3
2	STR X1
3	STR X0
4	
5	
6	
7	
8	

ORST

1	X1 OR (X2 AND X3)
2	STR X0
3	
⋮	
8	

AND X4

1	X4 AND [X1 OR (X2 AND X3)]
2	STR X0
3	
⋮	
8	

OR X5

1	X5 OR [X4 AND [X1 OR (X2 AND X3)]]
2	STR X0
3	
⋮	
8	

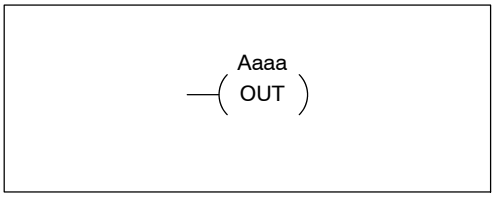
ANDST

1	X0 AND [(X5 OR [X4) AND [X1 OR (X2 AND X3)]]]
2	
3	
⋮	
8	

Out (OUT)

DS	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.

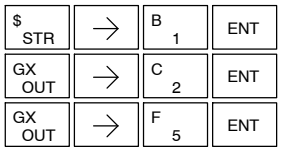
Operand Data Type		DL130 Range
	A	**aaa**
Inputs	X	0-11
Outputs	Y	0-7
Control Relays	C	0-377

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

DirectSOFT



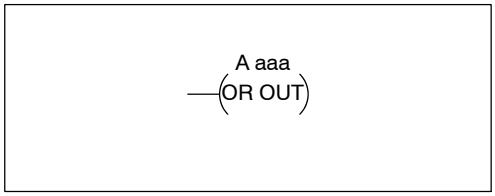
Handheld Programmer Keystrokes



Or Out (OR OUT)

DS	Used
HPP	Used

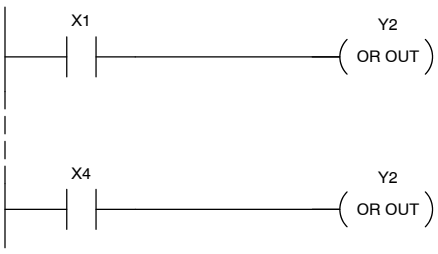
The Or Out instruction allows more than one rung of discrete logic to control a single output. Multiple Or Out instructions referencing the same output coil may be used, since *all* contacts controlling the output are logically ORed together. If the status of *any* rung is on, the output will also be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



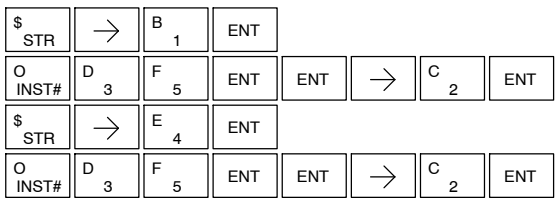
Operand Data Type		DL130 Range
	A	**aaa**
Inputs	X	0-177
Outputs	Y	0-177
Control Relays	C	0-377

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

DirectSOFT



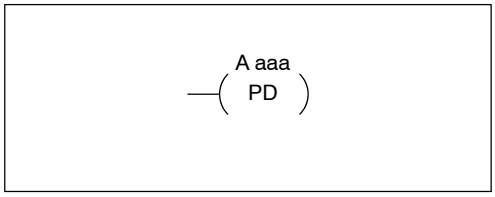
Handheld Programmer Keystrokes



Positive Differential (PD)

DS	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.



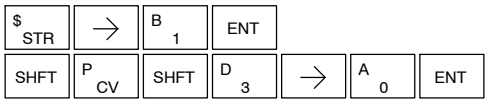
Operand Data Type		DL130 Range
	A	**aaa**
Inputs	X	0–11
Outputs	Y	0–7
Control Relays	C	0–377

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

DirectSOFT



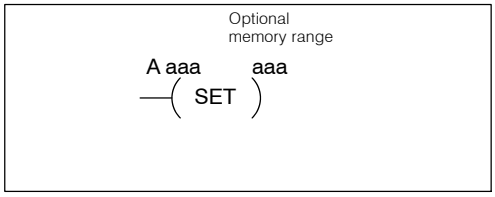
Handheld Programmer Keystrokes



Set (SET)

DS	Used
HPP	Used

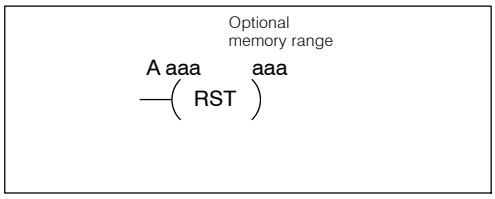
The Set instruction sets or turns on an image register point/memory location or a consecutive range of image register points/memory locations. Once the point/location is set it will remain on until it is reset using the Reset instruction. It is not necessary for the input controlling the Set instruction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



Reset (RST)

DS	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.



Operand Data Type		DL105 Range
	A	**aaa**
Inputs	X	0–11
Outputs	Y	0–7
Control Relays	C	0–377
Stage	S	0–377
Timer	T	0–77
Counter	CT	0–77

Set, Reset Instr. Continued

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

*Direct*SOFT



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
X SET	→	C 2	→	F 5	ENT			

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

*Direct*SOFT



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
S RST	→	C 2	→	F 5	ENT			

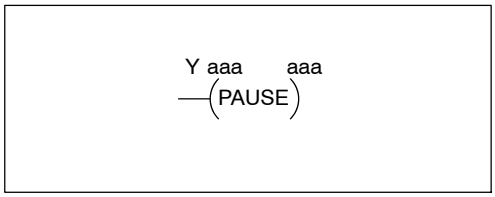
Pause (PAUSE)

DS	Used
HPP	Used

Handheld Programmer Keystrokes

$ STR	→	B 1	ENT							
O INST#	J 9	G 6	A 0	ENT	ENT	→	F 5	→	H 7	ENT

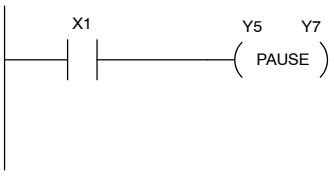
The Pause instruction disables the output update on a range of outputs. The ladder program will continue to run and update the image register. However, the outputs in the range specified in the Pause instruction will be turned off at the output points.



Operand Data Type	DL130 Range
	aaa
Outputs Y	0-7

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

*Direct*SOFT



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

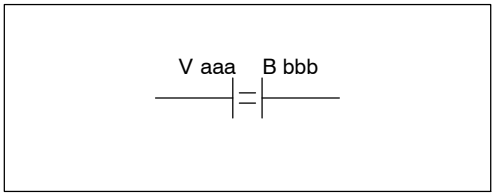
$ STR	→	B 1	ENT							
O INST#	J 9	G 6	A 0	ENT	ENT	→	F 5	→	H 7	ENT

In some cases, you may want certain output points in the specified pause range to operate normally. In that case, use Aux 58 to over-ride the Pause instruction.

Comparative Boolean

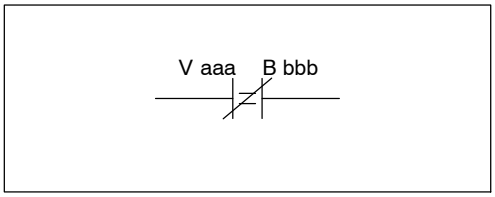
Store If Equal (STRE)

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



Store If Not Equal (STRNE)

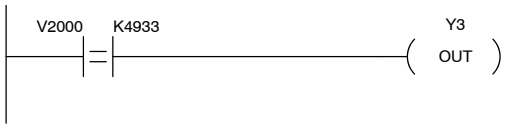
The Store If Not Equal instruction begins a new rung or additional branch in a rung with a normally closed comparative contact. The contact will be on when Vaaa ≠ Bbbb.



Operand Data Type		DL130 Range	
	B	**aaa**	**bbb**
V-memory	V	All (See page 4-29)	All (See page 4-29)
Constant	K	--	0-FFFF

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

DirectSOFT



Handheld Programmer Keystrokes

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0
→	E 4	J 9	D 3	D 3	ENT		
GX OUT	→	D 3	ENT				

In the following example, when the value in V-memory location V2000 ≠ 5060, Y3 will energize.

DirectSOFT

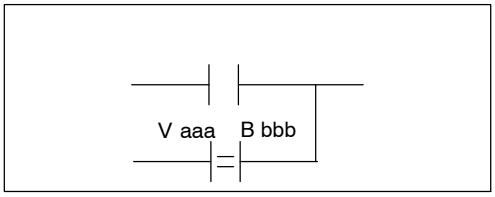


Handheld Programmer Keystrokes

SP STRN	SHFT	E 4	→	C 2	A 0	A 0	A 0
→	F 5	A 0	G 6	A 0	ENT		
GX OUT	→	D 3	ENT				

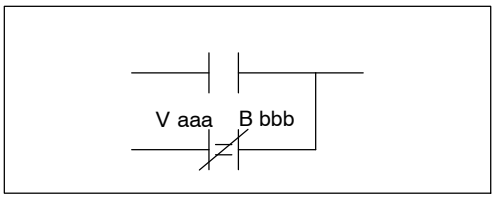
Or If Equal (ORE)

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



Or If Not Equal (ORNE)

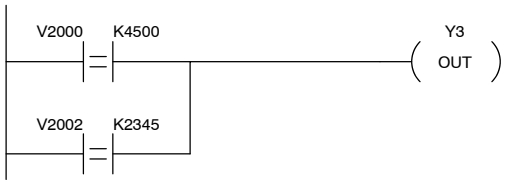
The Or If Not Equal instruction connects a normally closed comparative contact in parallel with another contact. The contact will be on when Vaaa ≠ Bbbb.



Operand Data Type		DL130 Range	
	B	**aaa**	**bbb**
V-memory	V	All (See page 4-29)	All (See page 4-29)
Constant	K	--	0-FFFF

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

DirectSOFT

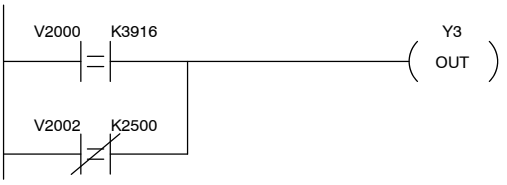


Handheld Programmer Keystrokes

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
E 4	F 5	A 0	A 0	ENT				
Q OR	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

In the following example, when the value in V-memory location V2000 = 3916 or V2002 ≠ 2500, Y3 will energize.

DirectSOFT

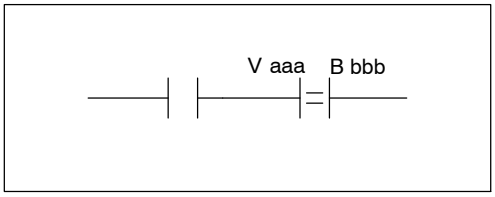


Handheld Programmer Keystrokes

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
D 3	J 9	B 1	G 6	ENT				
R ORN	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	D 3	ENT					

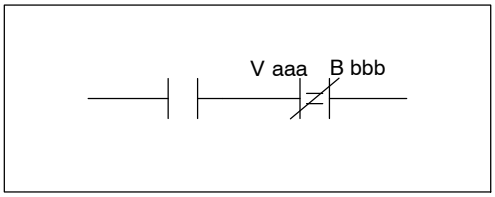
And If Equal (ANDE)

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



And If Not Equal (ANDNE)

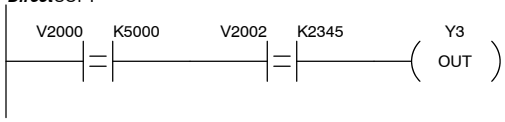
The And If Not Equal instruction connects a normally closed comparative contact in series with another contact. The contact will be on when Vaaa ≠ Bbbb



Operand Data Type		DL130 Range	
	A/B	aaa	bbb
V-memory	V	All (See page 4-29)	All (See page 4-29)
Constant	K	--	0-FFFF

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

DirectSOFT

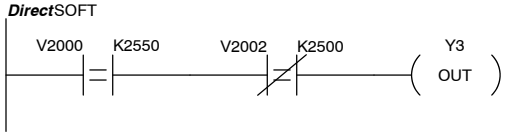


Handheld Programmer Keystrokes

$ 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 ≠ 2500, Y3 will energize.

DirectSOFT

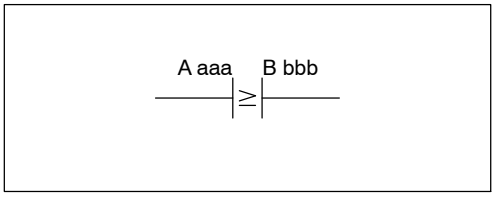


Handheld Programmer Keystrokes

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
C 2	F 5	F 5	A 0	ENT				
W ANDN	SHFT	E 4	→	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	D 3	ENT					

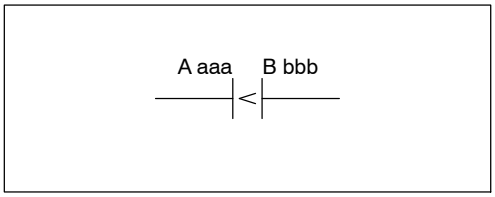
Store (STR)

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



Store Not (STRN)

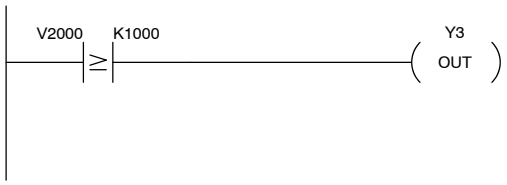
The Comparative Store Not instruction begins a new rung or additional branch in a rung with a normally closed comparative contact. The contact will be on when Aaaa < Bbbb.



Operand Data Type		DL130 Range	
	A/B	aaa	bbb
V-memory	V	All (See page 4-29)	All (See page 4-29)
Constant	K	--	0-FFFF
Timer	T	0-77	
Counter	CT	0-77	

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

***Direct*SOFT**



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.

***Direct*SOFT**

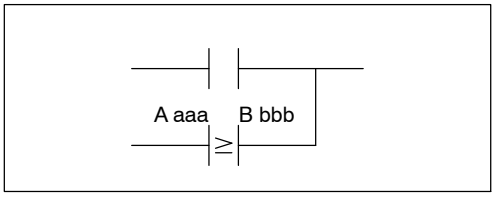


Handheld Programmer Keystrokes

SP STRN	→	SHFT	V AND	C 2	A 0	A 0	A 0
→	E 4	A 0	F 5	A 0	ENT		
GX OUT	→	D 3	ENT				

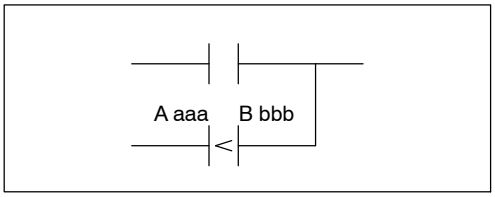
Or (OR)

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



Or Not (ORN)

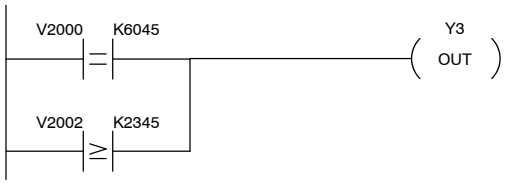
The Comparative Or Not instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa < Bbbb.



Operand Data Type		DL130 Range	
	A/B	aaa	bbb
V-memory	V	All (See page 4-29)	All (See page 4-29)
Constant	K	--	0-FFFF
Timer	T	0-77	
Counter	CT	0-77	

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

***Direct*SOFT**

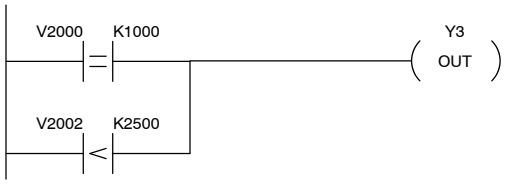


Handheld Programmer Keystrokes

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
G 6	A 0	E 4	F 5	ENT				
Q OR	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	D 3	E 4	F 5	ENT				
GX OUT	→	D 3	ENT					

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

***Direct*SOFT**

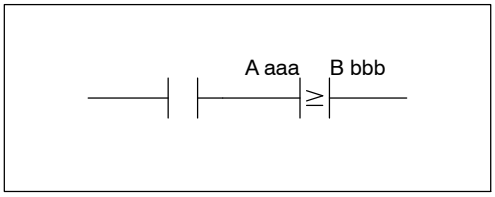


Handheld Programmer Keystrokes

$ STR	SHFT	E 4	→	C 2	A 0	A 0	A 0	→
B 1	A 0	A 0	A 0	ENT				
R ORN	→	SHFT	V AND	C 2	A 0	A 0	C 2	→
C 2	F 5	A 0	A 0	ENT				
GX OUT	→	D 3	ENT					

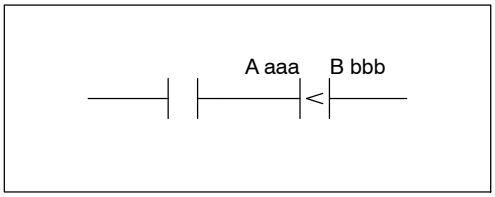
And (AND)

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



And Not (ANDN)

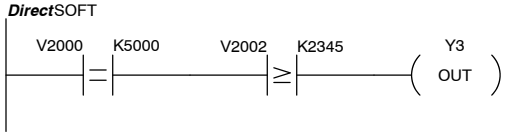
The Comparative And Not instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa < Bbbb.



Operand Data Type		DL130 Range	
	A/B	aaa	bbb
V-memory	V	All (See page 4-29)	All (See page 4-29)
Constant	K	--	0-FFFF
Timer	T	0-77	
Counter	CT	0-77	

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

DirectSOFT

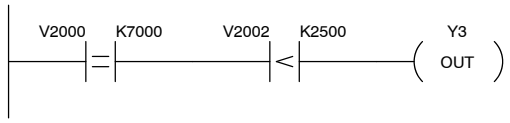


Handheld Programmer Keystrokes

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

In the following example, when the value in V-memory location V2000 = 7000 and V2002 < 2500, Y3 will energize.

DirectSOFT



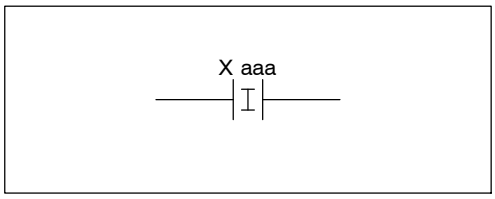
Handheld Programmer Keystrokes

$ 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	→	SHFT	Y AND	D 3	ENT			

Immediate Instructions

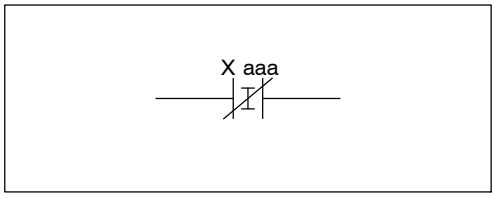
Store Immediate (STRI)

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



Store Not Immediate (STRNI)

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



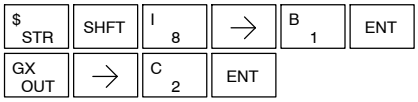
Operand Data Type		DL130 Range
		aaa
Inputs	X	0-11

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

DirectSOFT



Handheld Programmer Keystrokes

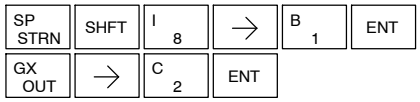


In the following example when X1 is off, Y2 will energize.

DirectSOFT

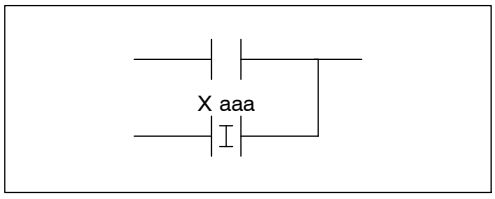


Handheld Programmer Keystrokes



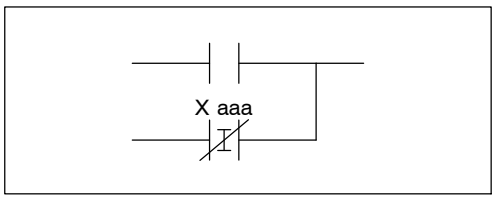
Or Immediate (ORI)

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



Or Not Immediate (ORNI)

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

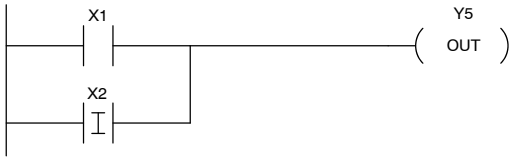


OR Immediate Instructions Cont'd

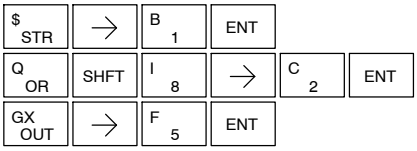
Operand Data Type		DL130 Range
		aaa
Inputs	X	0–177

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

*Direct*SOFT

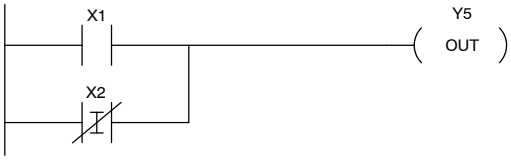


Handheld Programmer Keystrokes

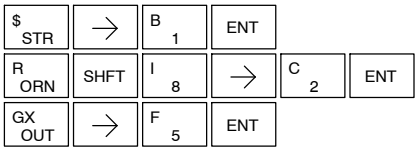


In the following example, when X1 is on or X2 is off, Y5 will energize.

*Direct*SOFT

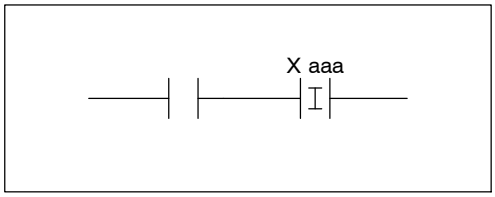


Handheld Programmer Keystrokes



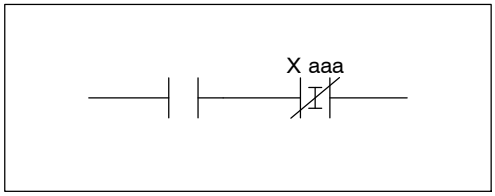
And Immediate (ANDI)

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



And Not Immediate (ANDNI)

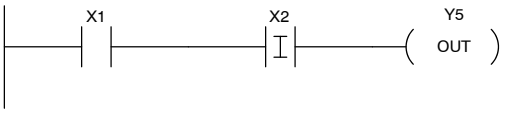
The And Not Immediate connects two contacts in series. The status of the contact will be opposite the status of the associated input point *at the time the instruction is executed*. The image register is not updated.



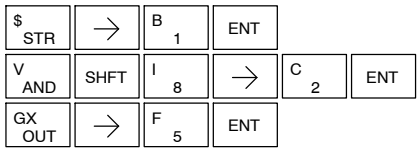
Operand Data Type		DL130 Range
		aaa
Inputs	X	0–11

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

*Direct*SOFT

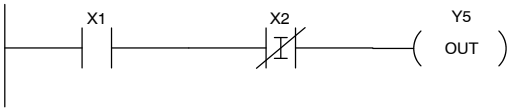


Handheld Programmer Keystrokes

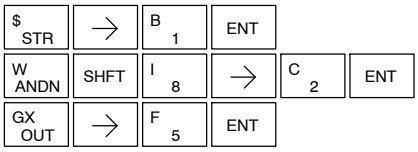


In the following example, when X1 is on and X2 are off, Y5 will energize.

*Direct*SOFT

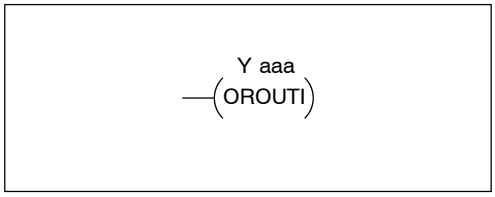


Handheld Programmer Keystrokes



Or Out Immediate (OROUTI)

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



Operand Data Type		DL130 Range
		aaa
Outputs	Y	0-177

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

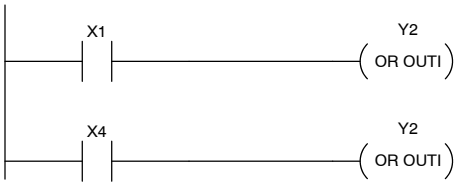


Handheld Programmer Keystrokes

$ STR	→	B 1	ENT			
O INST#	D 3	F 5	A 0	ENT	ENT	
→	C 2	ENT				

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

DirectSOFT

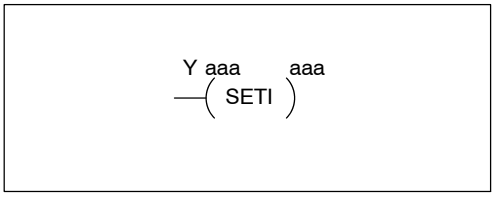


Handheld Programmer Keystrokes

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

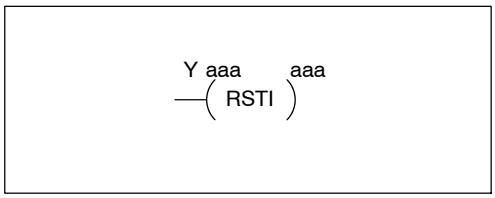
Set Immediate (SETI)

The Set Immediate instruction immediately sets, or turns on an output or a range of outputs in the image register and the corresponding output point(s) *at the time the instruction is executed*. Once the outputs are set it is not necessary for the input to remain on. The Reset Immediate instruction can be used to reset the outputs.



Reset Immediate (RSTI)

The Reset Immediate instruction immediately resets, or turns off an output or a range of outputs in the image register and the output point(s) *at the time the instruction is executed*. Once the outputs are reset it is not necessary for the input to remain on.



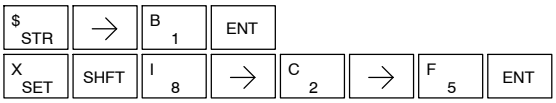
Operand Data Type		DL130 Range
		aaa
Outputs	Y	0-177

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.

***Direct*SOFT**

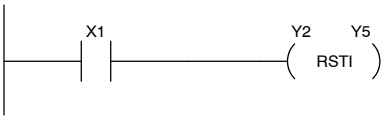


Handheld Programmer Keystrokes

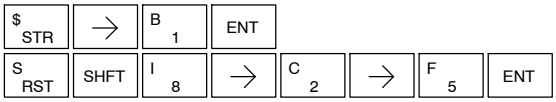


In the following example, when X1 is on, Y5 through Y22 will be reset (off) in the image register and on the corresponding output module(s).

***Direct*SOFT**



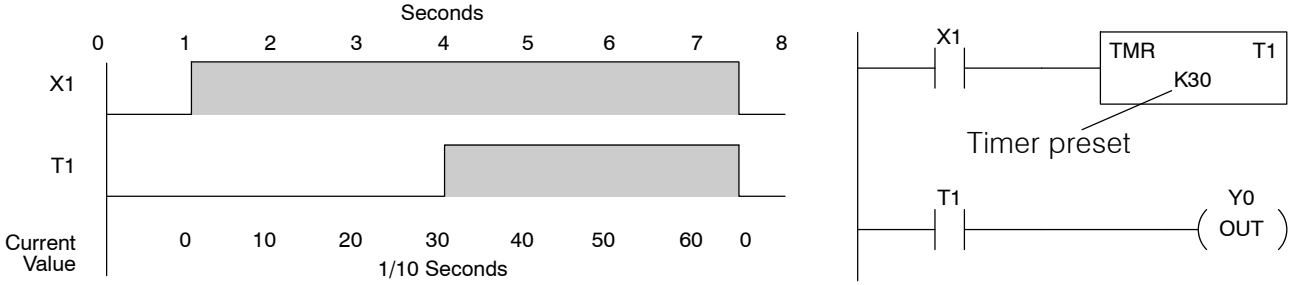
Handheld Programmer Keystrokes



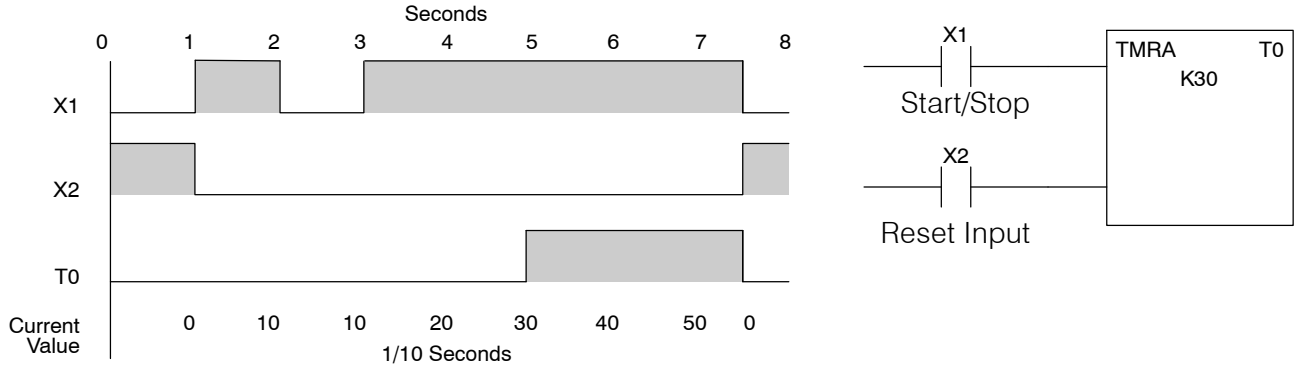
Timer, Counter and Shift Register Instructions

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)

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

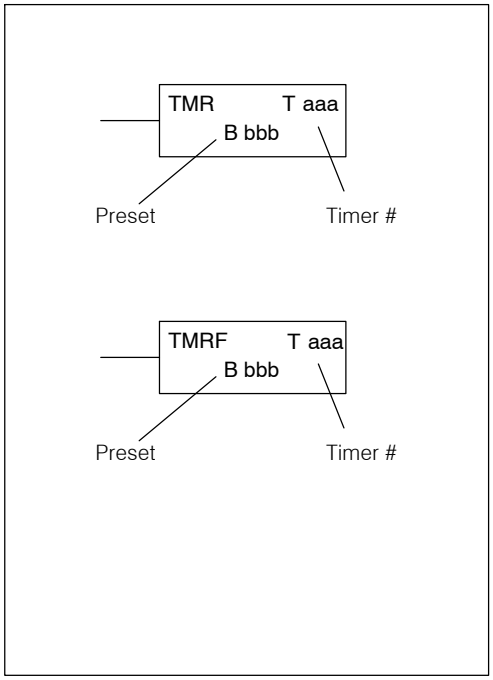
Instruction Specifications

Timer Reference (Taaa): Specifies the timer number.

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

Current Value: Timer current values are accessed by referencing the associated V or T memory location*. For example, the timer current value for T3 physically resides in V-memory location V3.

Discrete Status Bit: The discrete status bit is referenced by the associated T memory location. Operating as a "timer done bit", it will be on if the current value is equal to or greater than the preset value. For example, the discrete status bit for Timer 2 is TA2.



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

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		DL130 Range	
	A/B	aaa	bbb
Timers	T	0-77	--
V-memory for preset values	V	--	2000-2377 4000-4177
Constants (preset only)	K	--	0-9999
Timer discrete status bits	T/V	0-77 or V41100-41103	
Timer current values	V /T*	0-77	

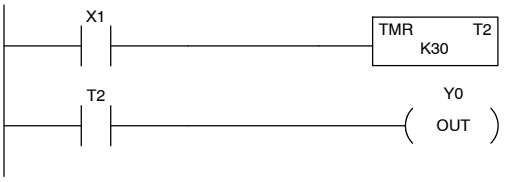
NOTE: * With the HPP, both the Timer discrete status bits and current value are accessed with the same data reference. ***Direct***SOFT uses separate references, such as "T2" for discrete status bit for Timer T2, and "TA2" for the current value of Timer T2.

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

Timer Example Using Discrete Status Bits

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

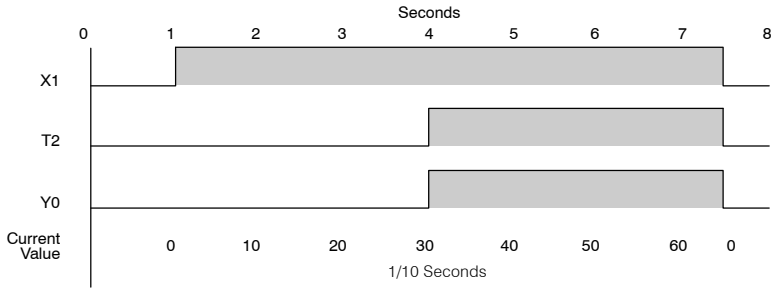
DirectSOFT



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							

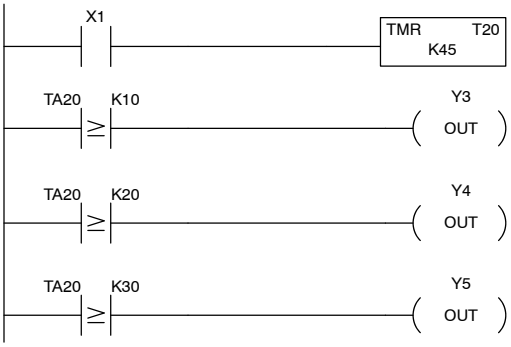
Timing Diagram



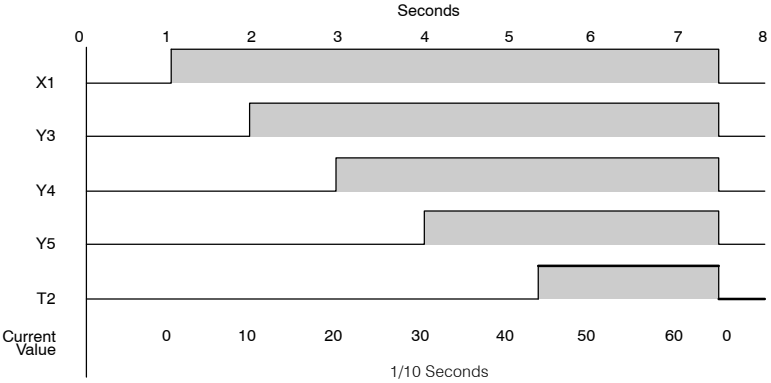
Timer Example Using Comparative Contacts

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



Timing Diagram



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)

Accumulating Fast Timer (TMRAF)

The Accumulating Timer is a 0.1 second two input timer that will time to a maximum of 9999999.9. The Accumulating Fast Timer is a 0.01 second two-input timer that will time to a maximum of 999999.99. *Each one uses two timer registers in V-memory.* These timers have two inputs, an enable and a reset. The timer starts timing when the enable is on and stops when the enable is off (without resetting the count). The reset will reset the timer when on and allow the timer to time when off.

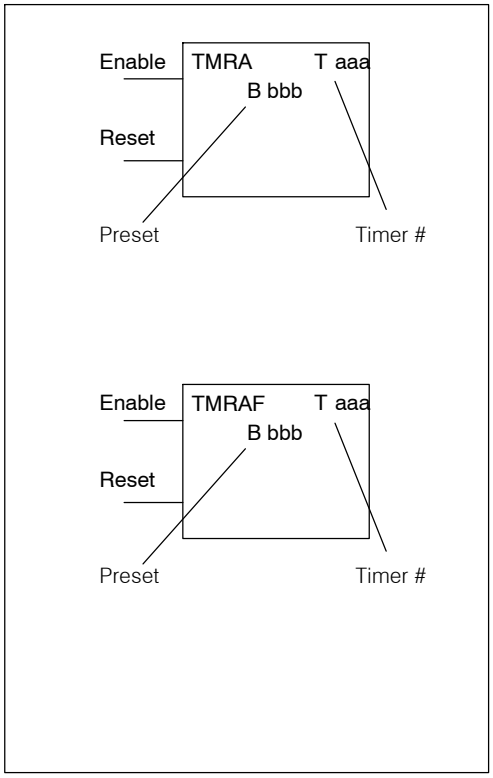
Instruction Specifications

Timer Reference (Taaa): Specifies the timer number.

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

Current Value: Timer current values are accessed by referencing the associated V or T memory location*. For example, the timer current value for T3 resides in V-memory location V3.

Discrete Status Bit: The discrete status bit is accessed by referencing the associated T memory location. Operating as a "timer done bit", it will be on if the current value is equal to or greater than the preset value. For example the discrete status bit for timer 2 would be T2.



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

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		DL130 Range	
	A/B	aaa	bbb
Timers	T	0-77	--
V-memory for preset values	V	--	2000-2376 4000-4176
Constants (preset only)	K	--	0-99999999
Timer discrete status bits	T/V	0-77 or V41100-41103	
Timer current values	V /T*	0-77	

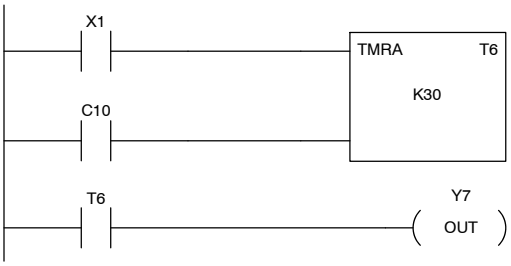
NOTE: * With the HPP, both the Timer discrete status bits and current value are accessed with the same data reference. ***Direct***SOFT uses separate references, such as T2 for discrete status bit for Timer T2, and TA2 for the current value of Timer T2.

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.

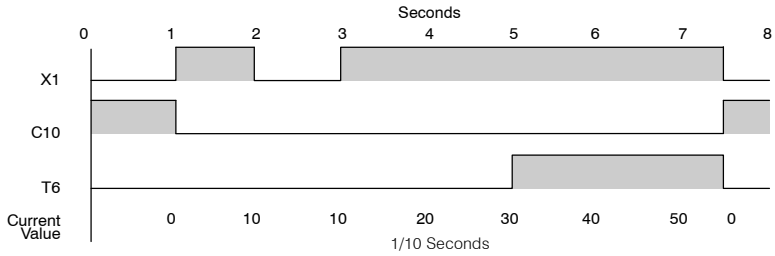
Accumulating Timer Example using Discrete Status Bits

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



Timing Diagram



Handheld Programmer Keystrokes

$ STR	→	B 1	ENT			
$ STR	→	SHFT	C 2	B 1	A 0	ENT
N TMR	SHFT	A 0	→	G 6	→	

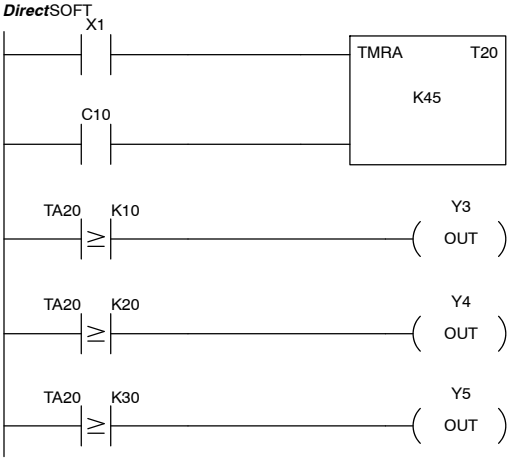
Handheld Programmer Keystrokes (cont)

D 3	A 0	ENT				
$ STR	→	SHFT	T MLR	G 6	ENT	
GX OUT	→	B 1	A 0	ENT		

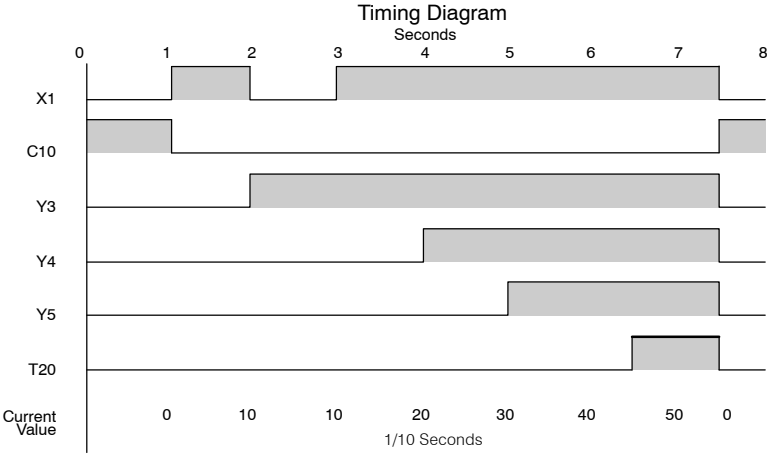
Accumulator Timer Example Using Comparative Contacts

In the following example, a 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.

DirectSOFT



Timing Diagram



Handheld Programmer Keystrokes

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

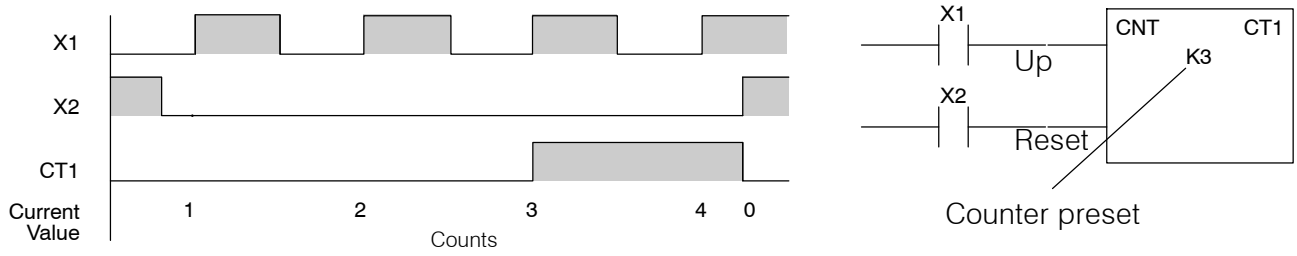
Handheld Programmer Keystrokes (cont)

GX OUT	→	E 4	ENT				
$ STR	→	SHFT	T MLR	C 2	A 0	→	
D 3	A 0	ENT					
GX OUT	→	F 5	ENT				

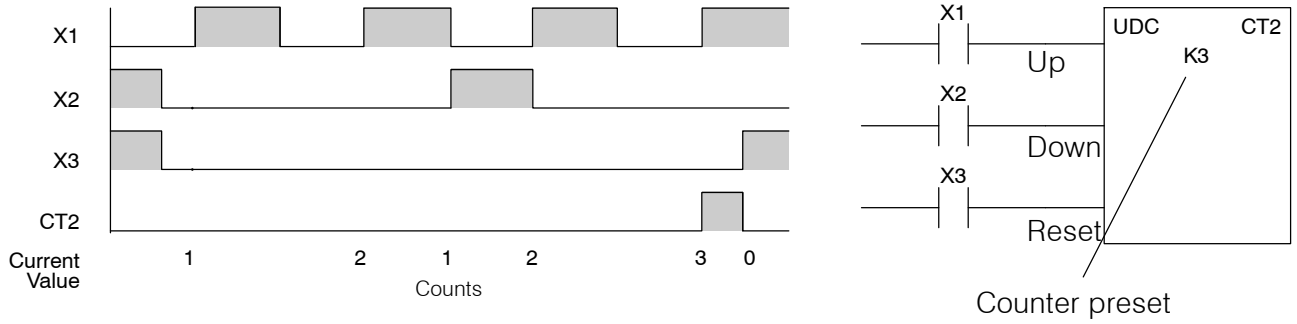
Using Counters

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

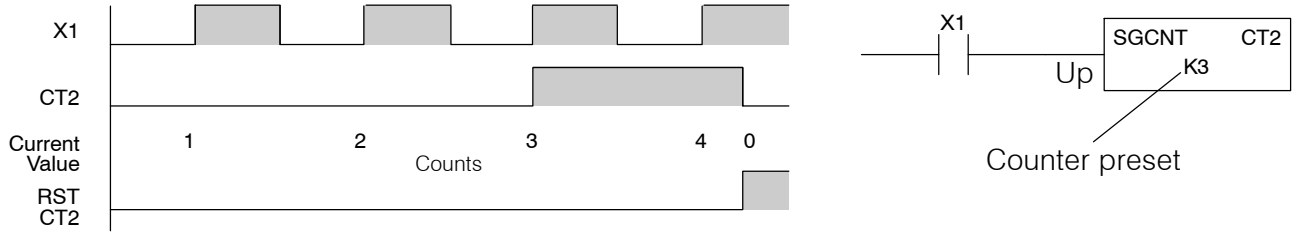
The up counter has two inputs, a count input and a reset input. The maximum count value is 9999. The timing diagram below shows the relationship between the counter input, counter reset, associated discrete bit, current value, and counter preset.



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

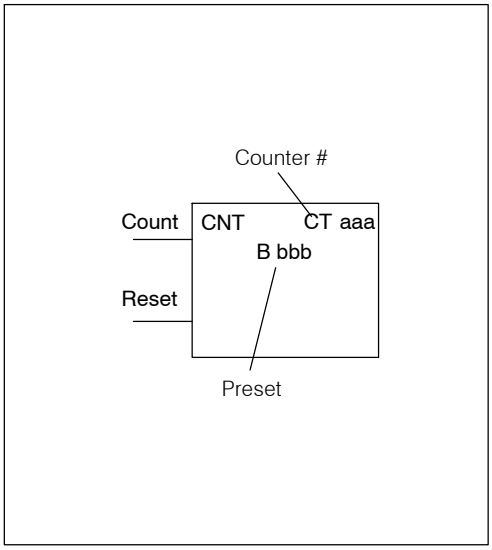


The stage counter has a count input and is reset by the RST instruction. This instruction is useful when programming using the RLL*PLUS* structured programming. The maximum count value is 9999. The timing diagram below shows the relationship between the counter input, associated discrete bit, current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



Counter (CNT)

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



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

Instruction Specifications

Counter Reference (CTaaa): Specifies the counter number.
Preset Value (Bbbb): Constant value (K) or a V–memory location.
Current Values: Counter current values are accessed by referencing the associated V or CT memory locations*. The V–memory location is the counter location + 1000. For example, the counter current value for CT3 resides in V–memory location V1003.

Discrete Status Bit: The discrete status bit is accessed by referencing the associated CT memory location. It will be on if the value is equal to or greater than the preset value. For example the discrete status bit for counter 2 would be CT2.



NOTE: Counter preset constants (K) may be changed by using a programming device, even when the CPU is in Run Mode. Therefore, a V-memory preset is required only if the ladder program must change the preset.

Operand Data Type		DL130 Range	
	A/B	aaa	bbb
Counters	CT	0–77	--
V–memory (preset only)	V	--	2000–2377 4000–4177
Constants (preset only)	K	--	0–9999
Counter discrete status bits	CT/V	0–77 or V41140–41143	
Counter current values	V/CT*	1000–1077	

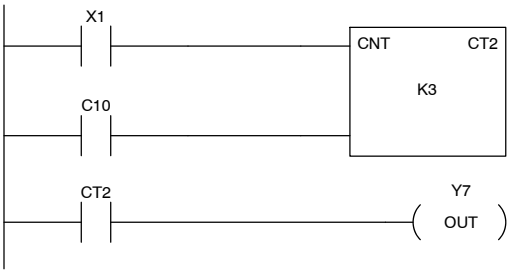


NOTE: * With the HPP, both the Counter discrete status bits and current value are accessed with the same data reference. ***Direct***SOFT uses separate references, such as “CT2” for discrete status bit for Counter CT2, and “CTA2” for the current value of Counter CT2.

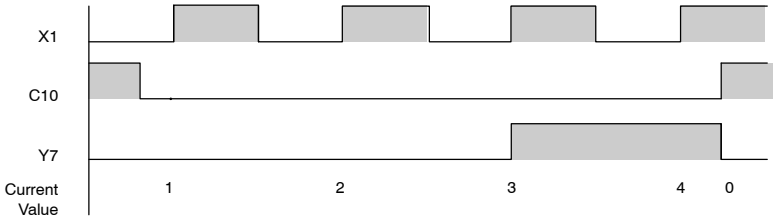
Counter Example Using Discrete Status Bits

In the following example, when X1 makes an off to on transition, counter CT2 will increment by one. When the current value reaches the preset value of 3, the counter status bit CT2 will turn on and energize Y7. When the reset C10 turns on, the counter status bit will turn off and the current value will be 0. The current value for counter CT2 will be held in V-memory location V1002.

***Direct*SOFT**



Counting diagram



Handheld Programmer Keystrokes

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

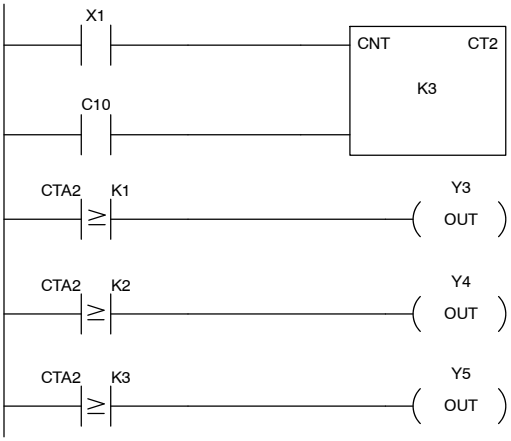
Handheld Programmer Keystrokes (cont)

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

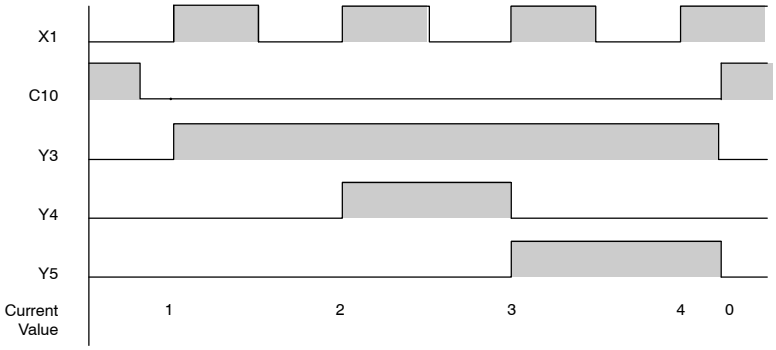
Counter Example Using Comparative Contacts

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.

***Direct*SOFT**



Counting diagram



Handheld Programmer Keystrokes

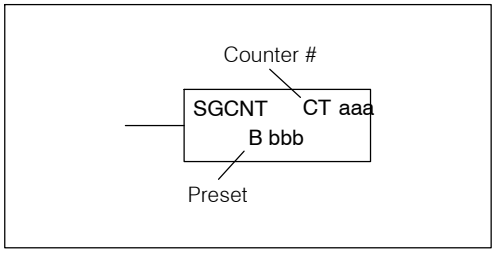
$ STR	→	B 1	ENT				
$ STR	→	SHFT	C 2	B 1	A 0	ENT	
GY CNT	→	C 2	→	D 3	ENT		
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2	
→	B 1	ENT					
GX OUT	→	D 3	ENT				

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)

The Stage Counter is a single input counter that increments when the input logic transitions from off to on. This counter differs from other counters since it will hold its current value until reset using the RST instruction. The Stage Counter is designed for use in RLL*PLUS* programs but can be used in relay ladder logic programs. When the current value equals the preset value, the counter status bit turns on and the counter continues to count up to a maximum count of 9999. The maximum value will be held until the counter is reset.



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

Instruction Specifications

Counter Reference (CTaaa): Specifies the counter number.
Preset Value (Bbbb): Constant value (K) or a V–memory location.
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.

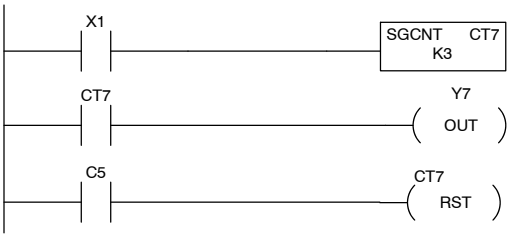
Operand Data Type		DL130 Range	
	A/B	aaa	bbb
Counters	CT	0–77	--
V–memory (preset only)	V	--	2000–2377
Constants (preset only)	K	--	0–9999
Counter discrete status bits	CT/V	0–77 or V41140–41143	
Counter current values	V/CT*	1000–1077	

NOTE: * With the HPP, both the Counter discrete status bits and current value are accessed with the same data reference. ***Direct***SOFT uses separate references, such as “CT2” for discrete status bit for Counter CT2, and “CTA2” for the current value of Counter CT2.

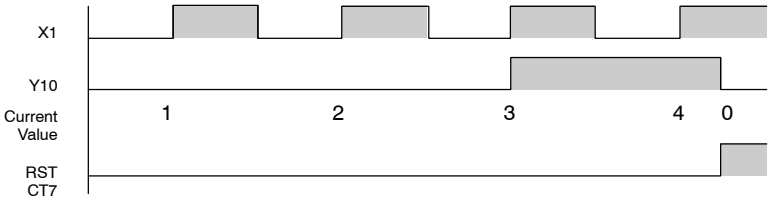
Stage Counter Example Using Discrete Status Bits

In the following example, when X1 makes an off to on transition, stage counter CT7 will increment by one. When the current value reaches 3, the counter status bit CT7 will turn on and energize Y7. The counter status bit CT7 will remain on until the counter is reset using the RST instruction. When the counter is reset, the counter status bit will turn off and the counter current value will be 0. The current value for counter CT7 will be held in V-memory location V1007.

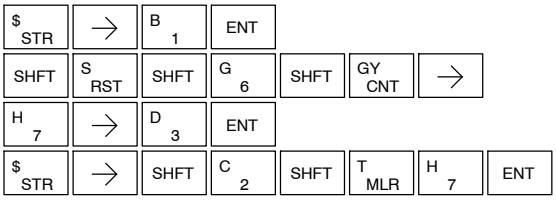
***Direct*SOFT**



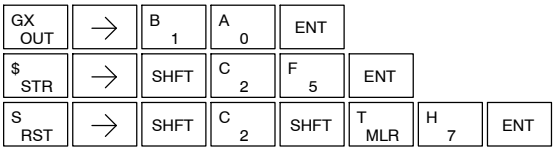
Counting diagram



Handheld Programmer Keystrokes



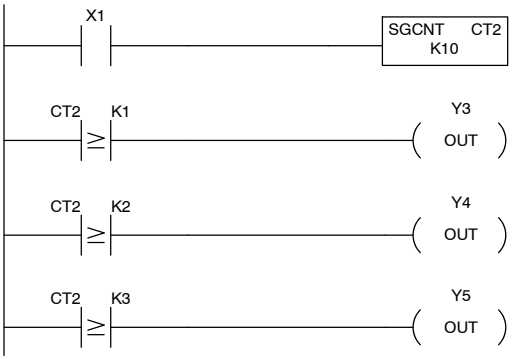
Handheld Programmer Keystrokes (cont)



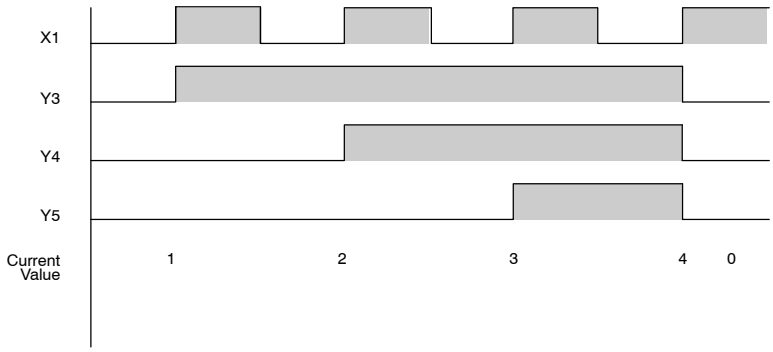
Stage Counter Example Using Comparative Contacts

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.

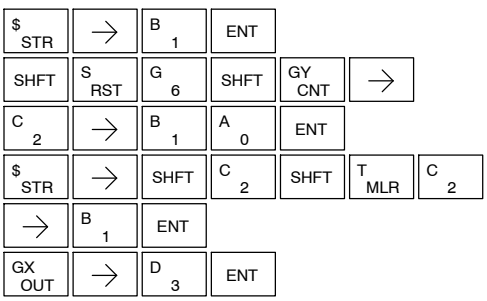
***Direct*SOFT**



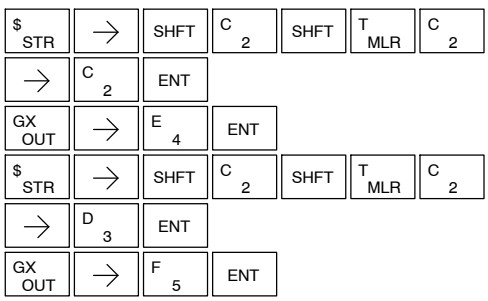
Counting diagram



Handheld Programmer Keystrokes

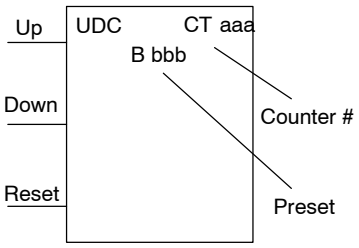


Handheld Programmer Keystrokes (cont)



Up Down Counter (UDC)

This Up/Down Counter counts up on each off to on transition of the Up input and counts down on each off to on transition of the Down input. The counter is reset to 0 when the Reset input is on. The count range is 0–99999999. The count input not being used must be off in order for the active count input to function.



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

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

Instruction Specification

Counter Reference (CTaaa): Specifies the counter number.
Preset Value (Bbbb): Constant value (K) or two consecutive V–memory locations.
Current Values: Current count is a double word value accessed by referencing the associated V or CT memory locations*. The V-memory location is the counter location + 1000. For example, the counter current value for CT5 resides in V–memory location V1005 and V1006.
Discrete Status Bit: The discrete status bit is accessed by referencing the associated CT memory location. Operating as a “counter done bit” it will be on if the value is equal to or greater than the preset value. For example the discrete status bit for counter 2 would be CT2.

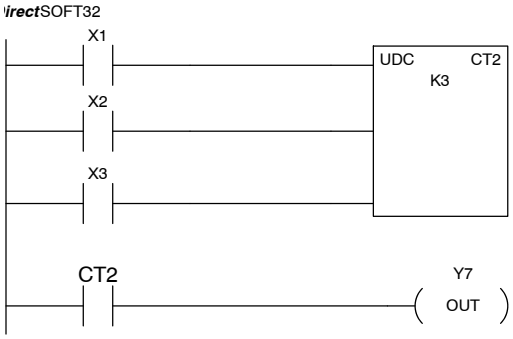
Operand Data Type		DL130 Range	
	A/B	aaa	bbb
Counters	CT	0–77	--
V–memory (preset only)	V	--	2000–2377 4000–4177
Constants (preset only)	K	--	0–99999999
Counter discrete status bits	CT/V	0–77 or V41140–41143	
Counter current values	V/CT*	1000–1077	

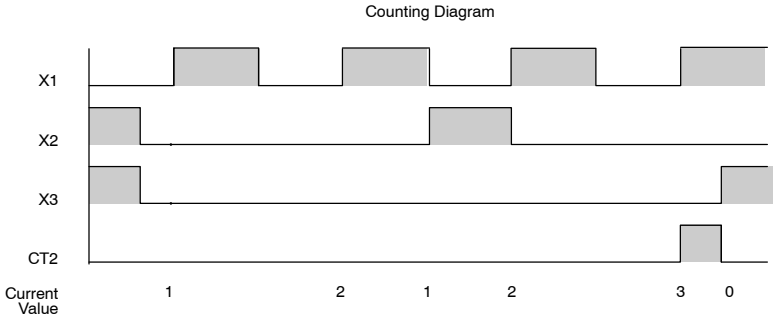


NOTE: * With the HPP, both the Counter discrete status bits and current value are accessed with the same data reference. ***Direct***SOFT uses separate references, such as “CT2” for discrete status bit for Counter CT2, and “CTA2” for the current value of Counter CT2.

Up / Down Counter Example Using Discrete Status Bits

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





Handheld Programmer Keystrokes

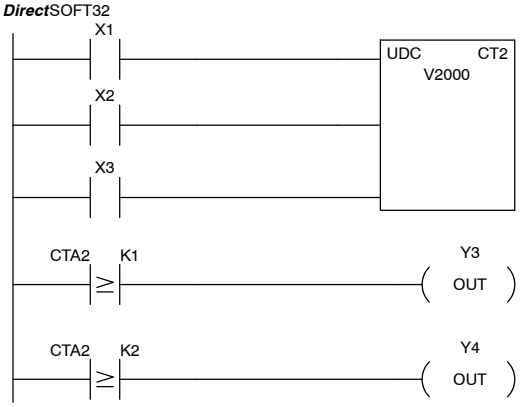
$ STR	→	B 1	ENT			
$ STR	→	C 2	ENT			
$ STR	→	D 3	ENT			
SHFT	U ISG	D 3	C 2	→	C 2	

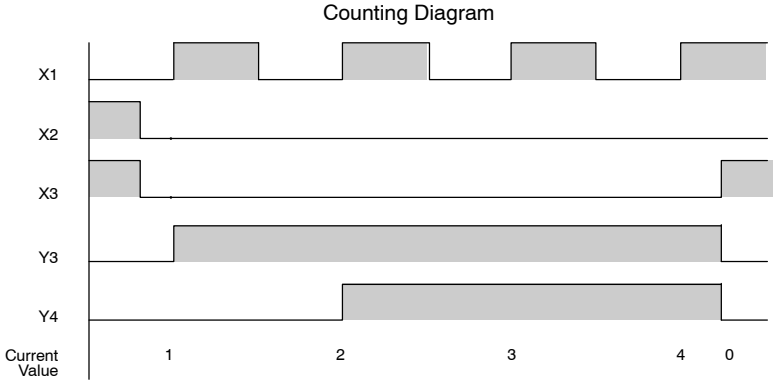
Handheld Programmer Keystrokes (cont)

→	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.





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)

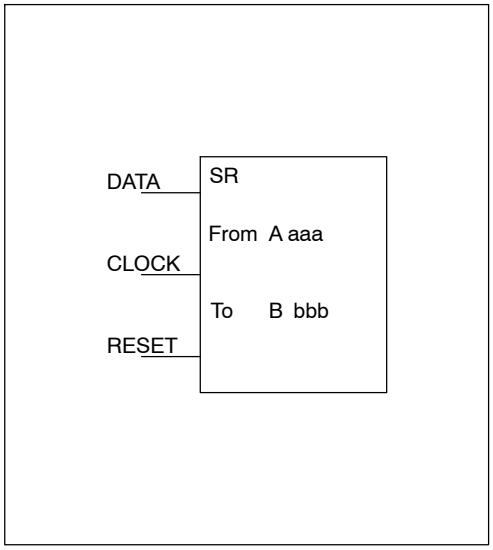
→	B 1	ENT				
GX OUT	→	D 3	ENT			
$ STR	→	SHFT	C 2	SHFT	T MLR	C 2
→	C 2	ENT				
GX OUT	→	E 4	ENT			

Shift Register (SR)

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.

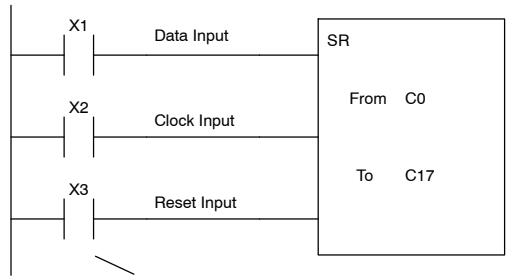
- Data — determines the value (1 or 0) that will enter the register
- Clock — shifts the bits one position on each low to high transition
- Reset —resets the Shift Register to all zeros.



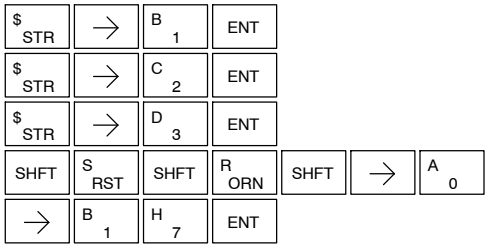
With each off to on transition of the clock input, the bits which make up the shift register block are shifted by one bit position and the status of the data input is placed into the starting bit position in the shift register. The direction of the shift depends on the entry in the From and To fields. 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		DL130 Range	
	A/B	aaa	bbb
Control Relay	C	0–377	0–377

DirectSOFT32



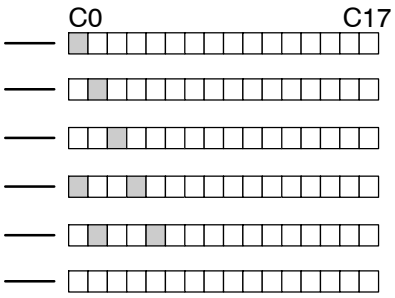
Handheld Programmer Keystrokes



Inputs on Successive Scans

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

Shift Register Bits



■ - indicates on □ - indicates off

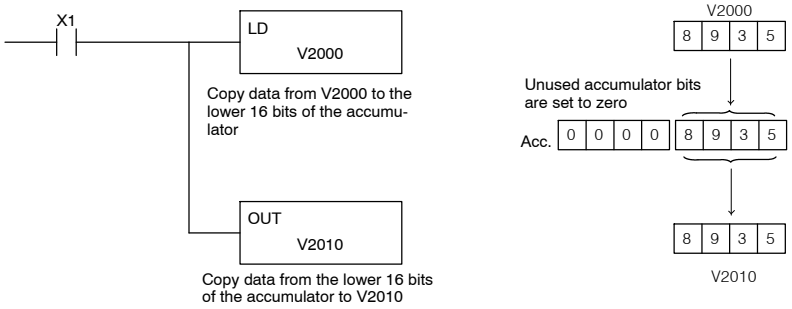
Accumulator / Stack Load and Output Data Instructions

Using the Accumulator

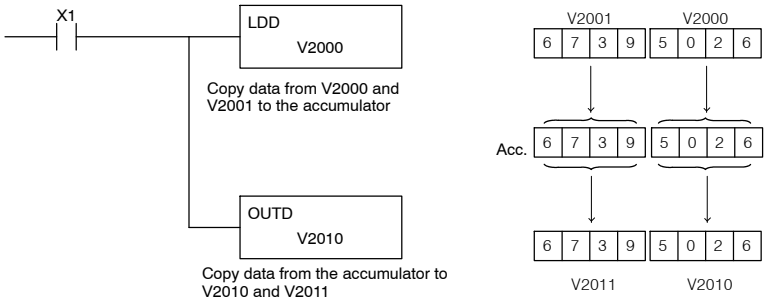
The accumulator in the DL105 internal CPUs is a 32 bit register which is used as a temporary storage location for data that is being copied or manipulated in some manor. For example, you have to use the accumulator to perform math operations such as add, subtract, multiply, etc. Since there are 32 bits, 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.

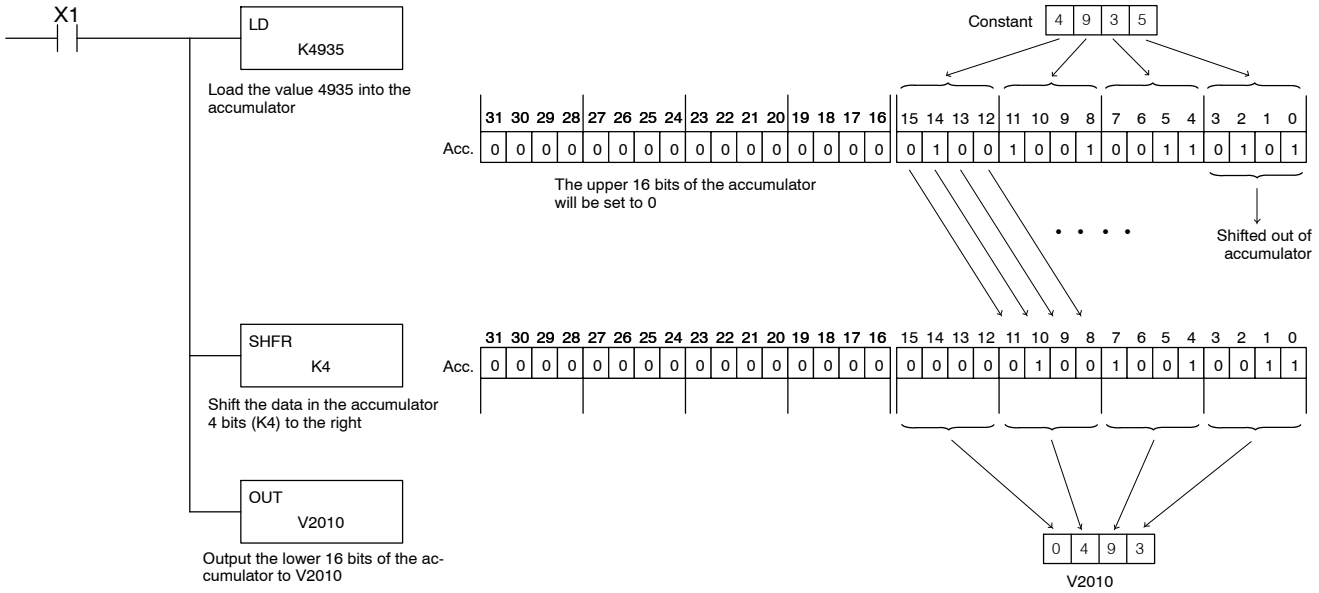


Since the accumulator is 32 bits and V memory locations are 16 bits the Load Double and Out Double (or variations thereof) use two consecutive V-memory locations or 8 digit BCD constants to copy data either to the accumulator from a V-memory address or from a V-memory address to the accumulator. For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



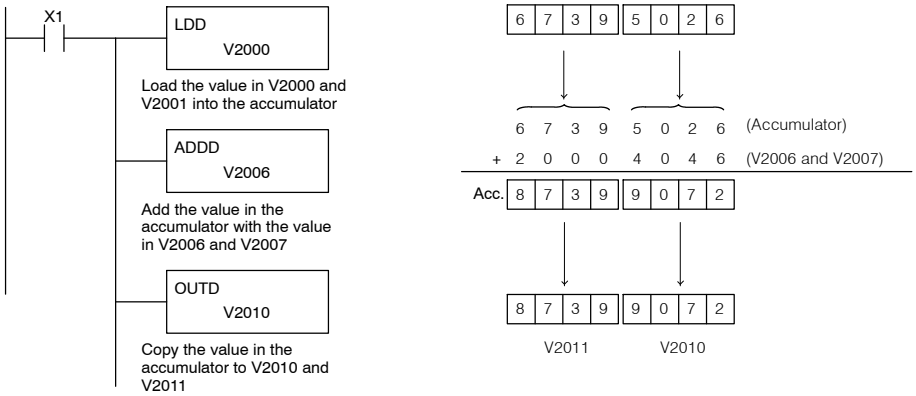
Changing the Accumulator Data

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.



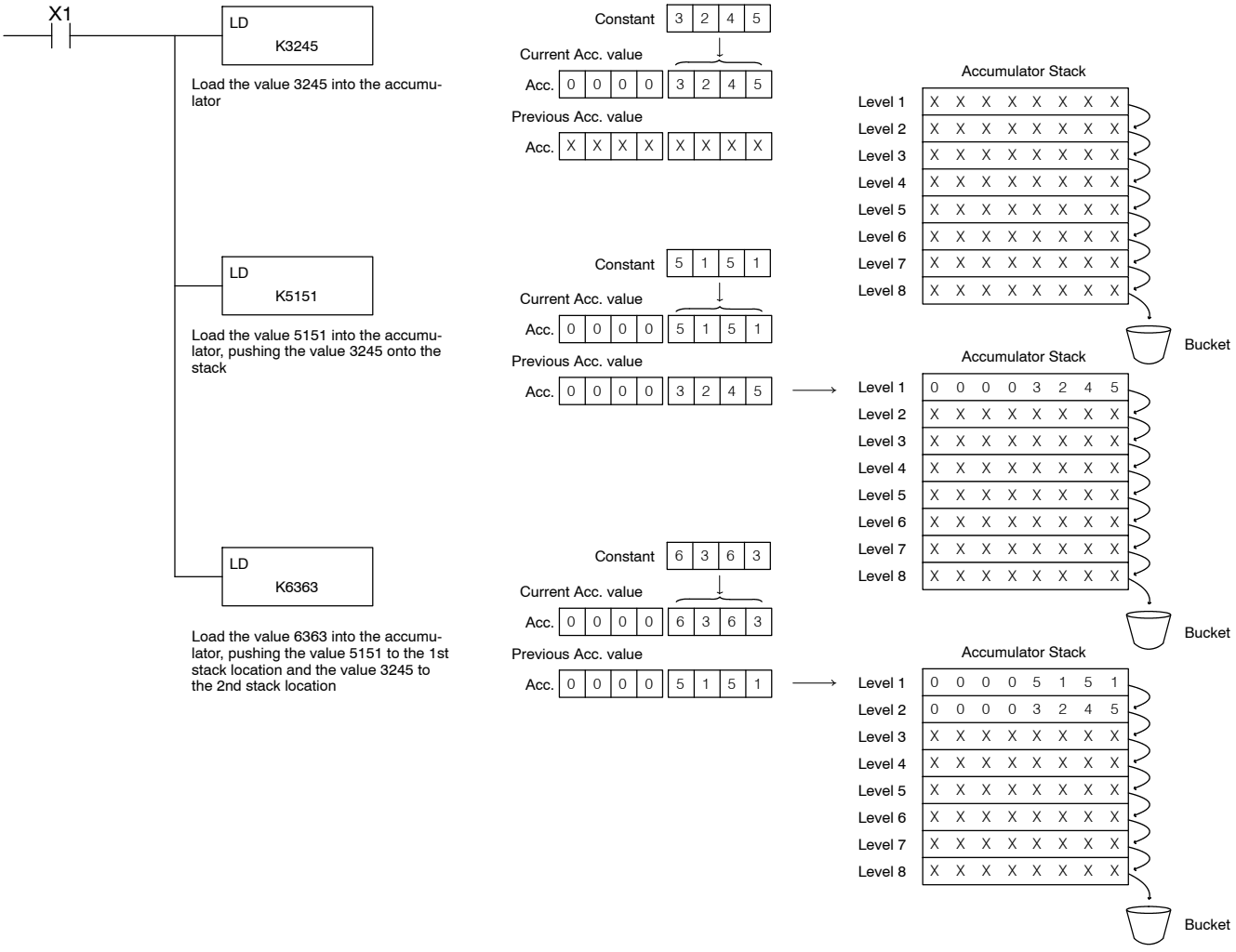
Some of the data manipulation instructions use 32 bits. They use two consecutive V memory locations or an 8 digit BCD constant to manipulate data in the accumulator.

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

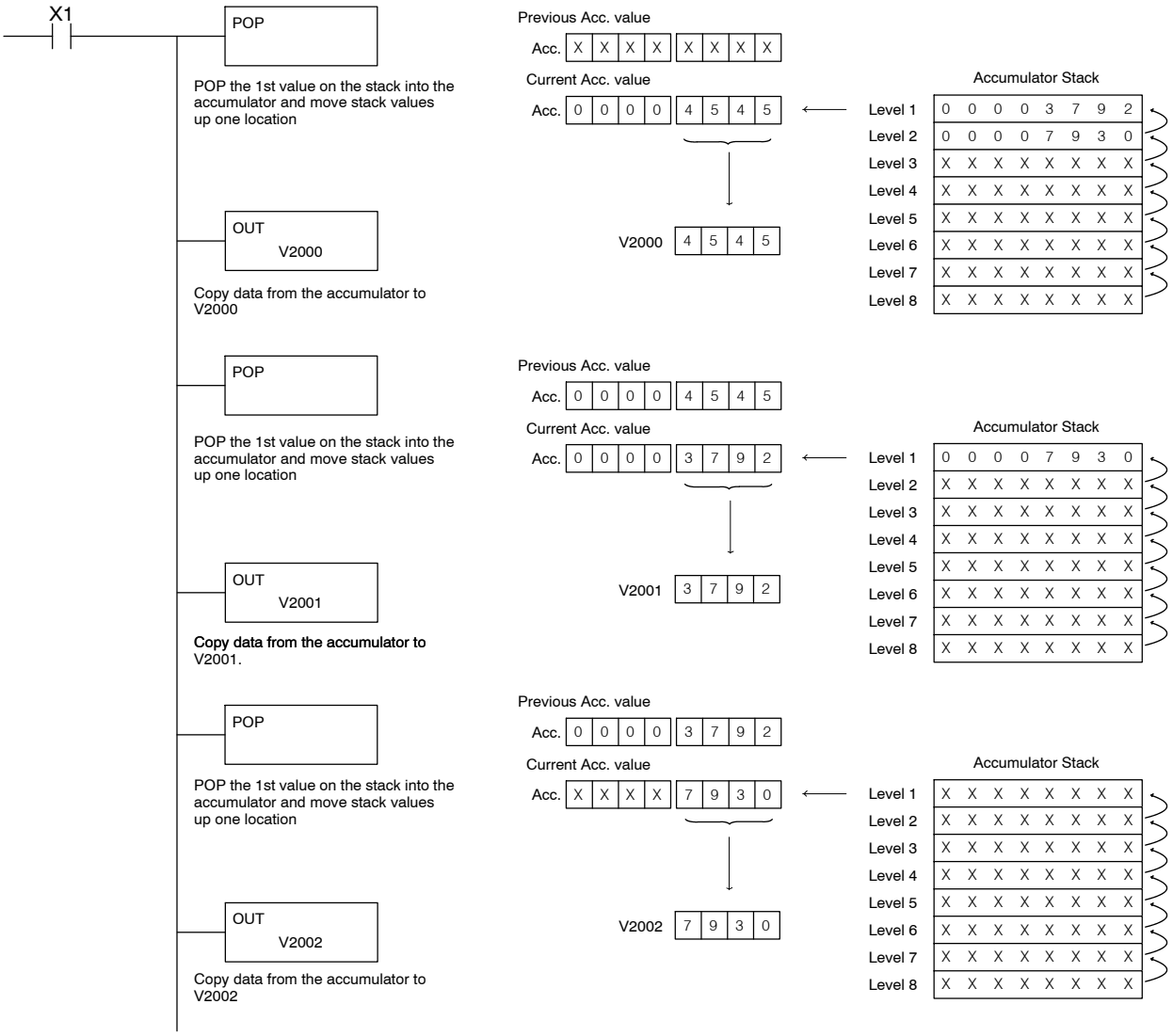


Using the Accumulator Stack

The accumulator stack is used for instructions that require more than one parameter to execute a function or for user defined functionality. The accumulator stack is used when more than one Load instruction is executed without the use of an Out instruction. The first load instruction in the scan places a value into the accumulator. Every Load instruction thereafter without the use of an Out instruction places a value into the accumulator and the value that was in the accumulator is placed onto the accumulator stack. The Out instruction nullifies the previous load instruction and does not place the value that was in the accumulator onto the accumulator stack when the next load instruction is executed. Every time a value is placed onto the accumulator stack the other values in the stack are pushed down one location. The accumulator is eight levels deep (eight 32 bit registers). If there is a value in the eighth location when a new value is placed onto the stack, the value in the eighth location is pushed off the stack and cannot be recovered.



The POP instruction rotates values upward through the stack into the accumulator. When a POP is executed the value which was in the accumulator is cleared and the value that was on top of the stack is in the accumulator. The values in the stack are shifted up one position in the stack.



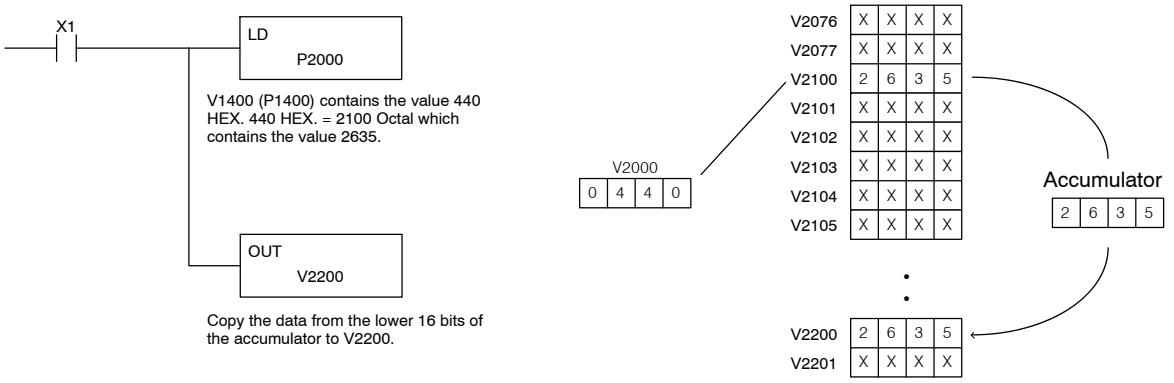
Using Pointers

Many of the DL105 series instructions will allow V-memory pointers as a operand (commonly known as indirect addressing). Pointers allow instructions to obtain data from V-memory locations referenced by the pointer value.

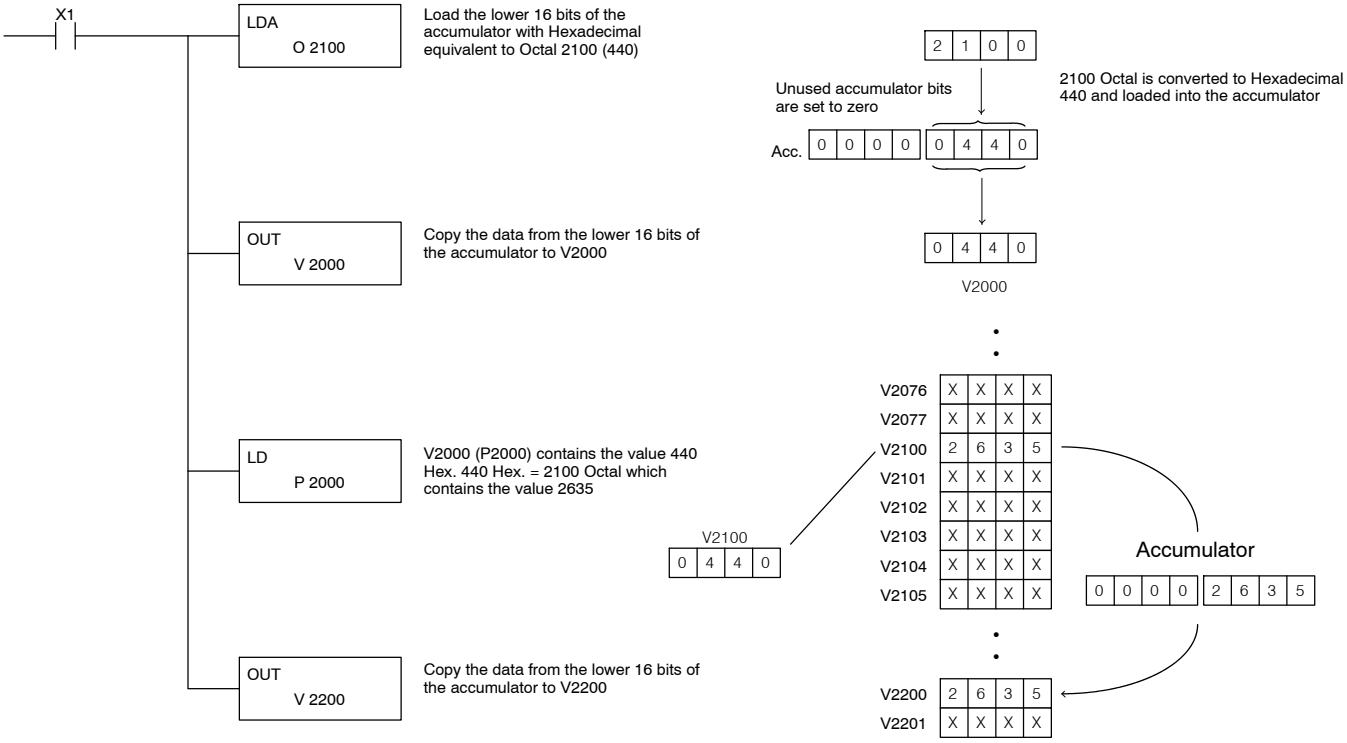


NOTE: DL105 V-memory addressing is in octal. However, the pointers reference a V-memory location with values viewed as HEX. Use the Load Address (LDA) instruction to move an address into the pointer location. This instruction performs the Octal to Hexadecimal conversion automatically.

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

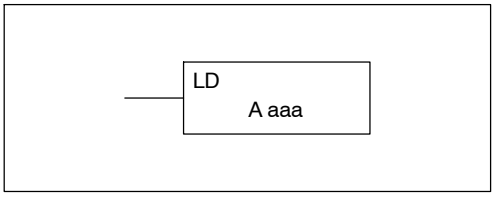


The following example is identical to the one above with one exception. The LDA (Load Address) instruction automatically converts the Octal address to Hex.



Load (LD)

The Load instruction is a 16 bit instruction that loads the value (Aaaa), which is either a V-memory location or a 4 digit constant, into the lower 16 bits of the accumulator. The upper 16 bits of the accumulator are set to 0.



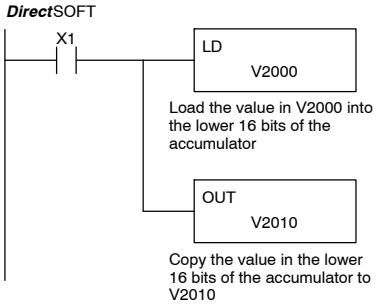
Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)
Pointer	P	All V mem. (See page 4-29)
Constant	K	0-FFFF

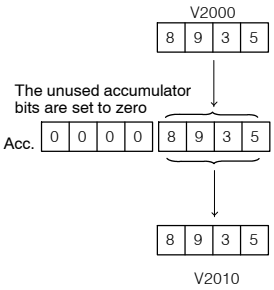
Discrete Bit Flags	Description
SP76	on when the value loaded into the accumulator by any instruction is zero.



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

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



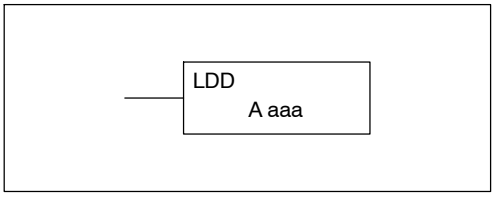


Handheld Programmer Keystrokes

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

Load Double (LDD)

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



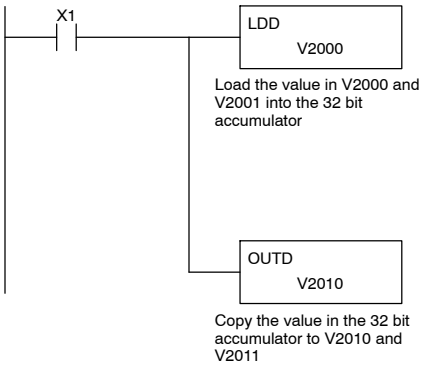
Operand Data Type		DL230 Range
	A	aaa
V-memory	V	All (See page 4-29)
Pointer	P	All V mem. (See page 4-29)
Constant	K	0-FFFFFFFF

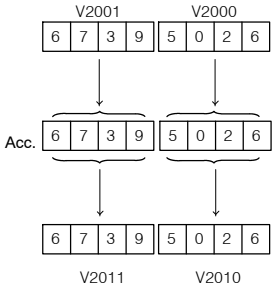
Discrete Bit Flags	Description
SP76	on when the value loaded into the accumulator by any instruction is zero.

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

In the following example, when X1 is on, the 32 bit value in V2000 and V2001 will be loaded into the accumulator and output to V2010 and V2011.

***Direct*SOFT**



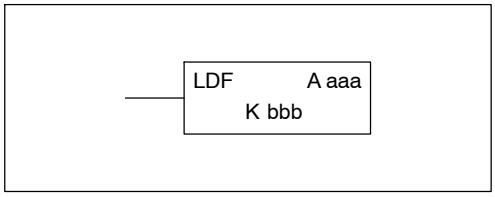


Handheld Programmer Keystrokes

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

Load Formatted (LDF)

The Load Formatted instruction loads 1-32 consecutive bits from discrete memory locations into the accumulator. The instruction requires a starting location (Aaaa) and the number of bits (Kbbb) to be loaded. Unused accumulator bit locations are set to zero.



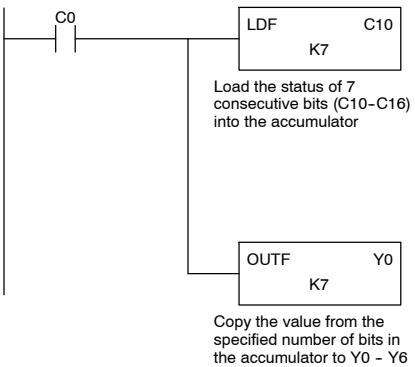
Operand Data Type		DL130 Range	
	A	**aaa**	**bbb**
Inputs	X	0-11	--
Outputs	Y	0-7	--
Control Relays	C	0-377	--
Stage Bits	S	0-377	--
Timer Bits	T	0-77	--
Counter Bits	CT	0-77	--
Special Relays	SP	0-117 540-577	--
Constant	K	--	1-32

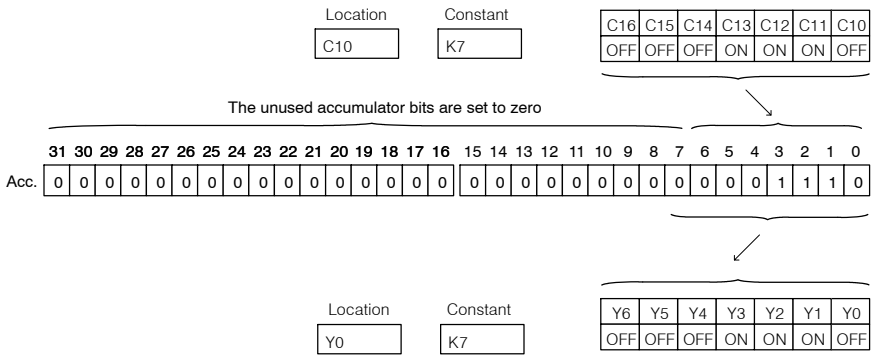
Discrete Bit Flags	Description
SP76	on when the value loaded into the accumulator by any instruction is zero.

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

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

***Direct*SOFT**



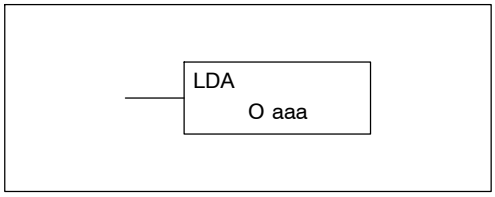


Handheld Programmer Keystrokes

$ 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)

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 DL105 system are in octal.



Operand Data Type		DL130 Range
		aaa
Octal Address	O	All V mem. (See page 4-29)

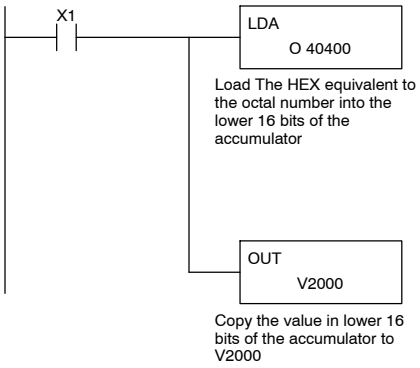
Discrete Bit Flags	Description
SP76	on when the value loaded into the accumulator by any instruction is zero.

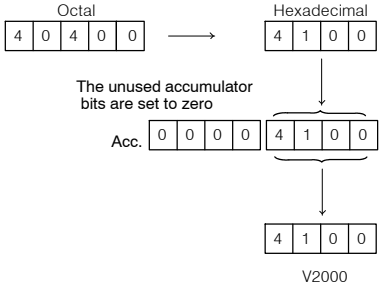


NOTE: Two consecutive Load instructions will place the value of the first load instruction onto the accumulator stack.

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

***Direct*SOFT**





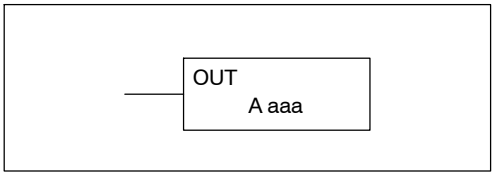
Handheld Programmer Keystrokes

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

Out (OUT)

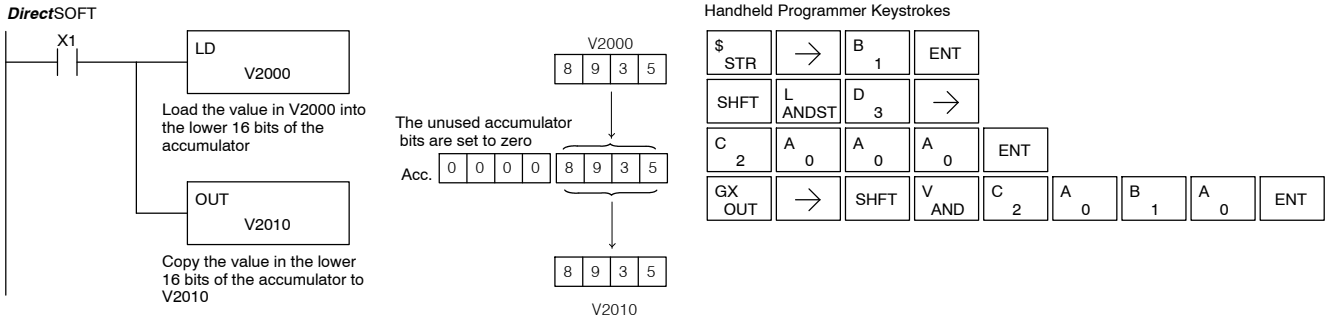
The Out instruction is a 16 bit instruction that copies the value in the lower 16 bits of the accumulator to a specified V-memory location (Aaaa).

Note: See Appendix E



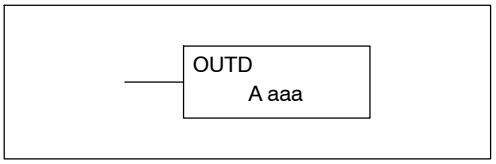
Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)
Pointer	P	All V mem. (See page 4-29)

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.



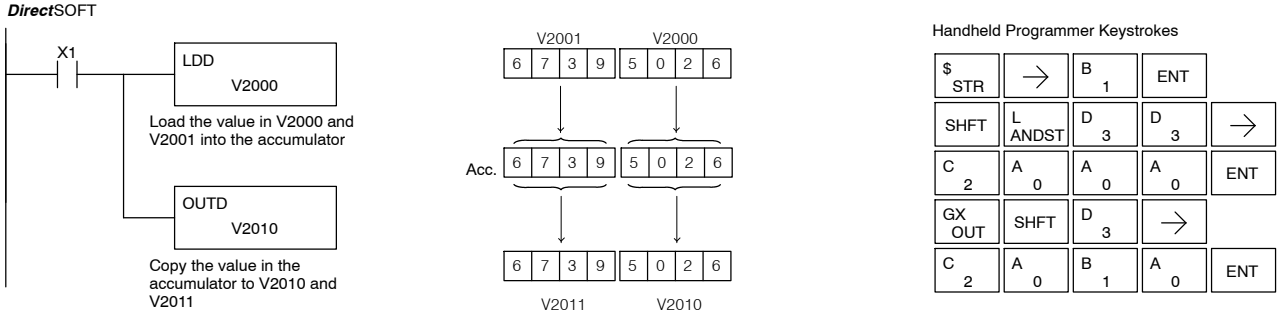
Out Double (OUTD)

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



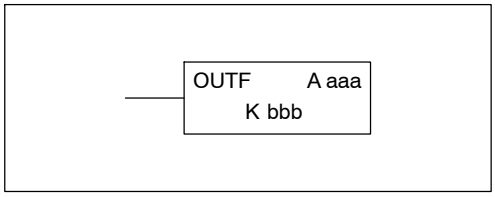
Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)
Pointer	P	All V mem. (See page 4-29)

In the following example, when X1 is on, the 32 bit value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.



Out Formatted (OUTF)

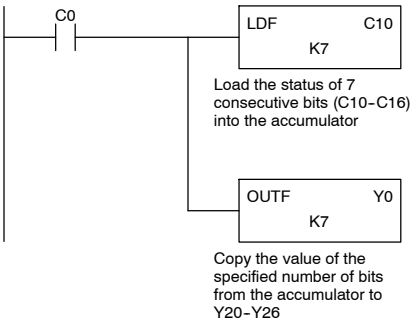
The Out Formatted instruction outputs 1-32 bits from the accumulator to the specified discrete memory locations. The instruction requires a starting location (Aaaa) for the destination and the number of bits (Kbbb) to be output.

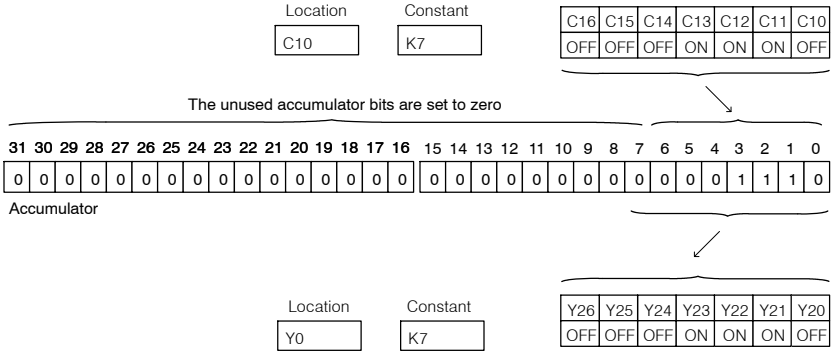


Operand Data Type		DL130 Range	
	A	aaa	bbb
Inputs	X	0-77	--
Outputs	Y	0-77	--
Control Relays	C	0-377	--
Constant	K	--	1-32

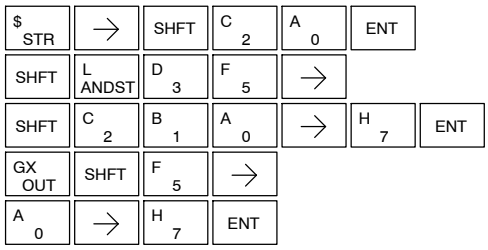
In the following example, when C0 is on, the binary pattern of C10-C16 (7 bits) will be loaded into the accumulator using the Load Formatted instruction. The lower 7 bits of the accumulator are output to Y0-Y6 using the Out Formatted instruction.

***Direct*SOFT**



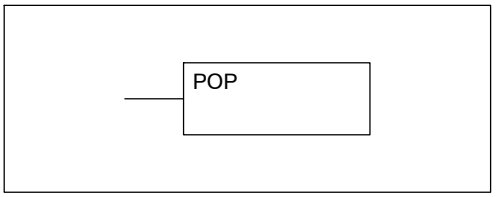


Handheld Programmer Keystrokes



Pop (POP)

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

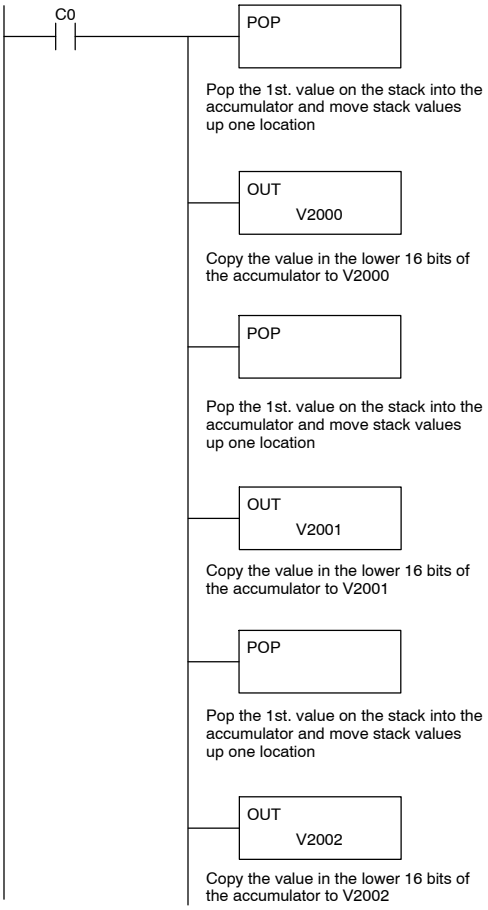


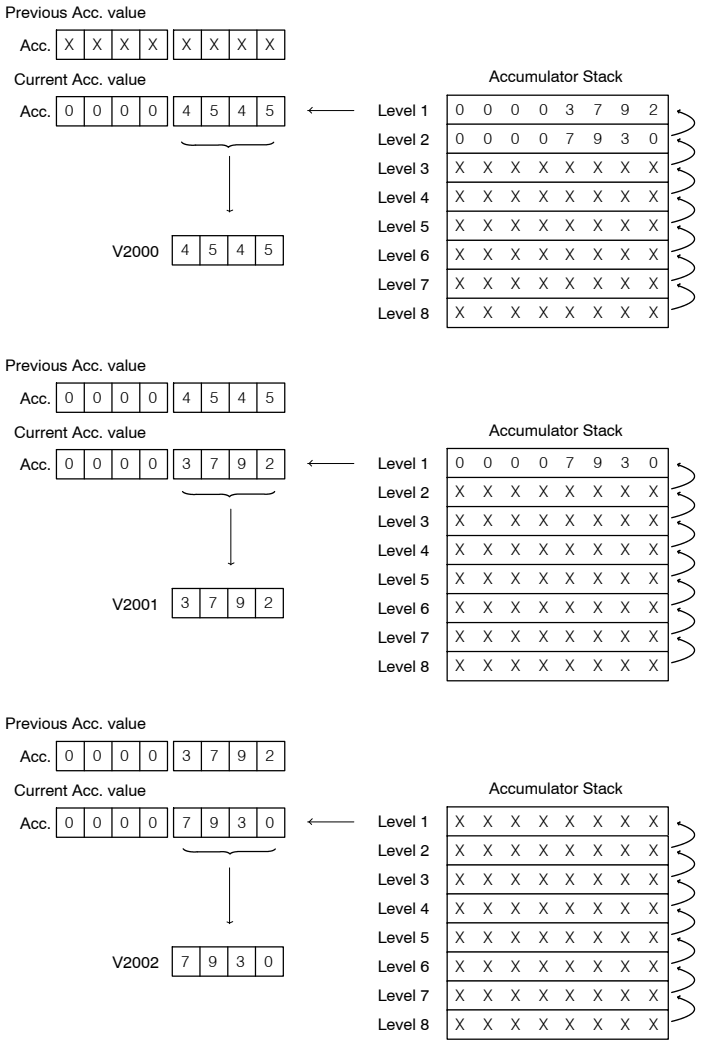
Discrete Bit Flags	Description
SP63	on when the result of the instruction causes the value in the accumulator to be zero.

Pop Instruction Continued

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

DirectSOFT





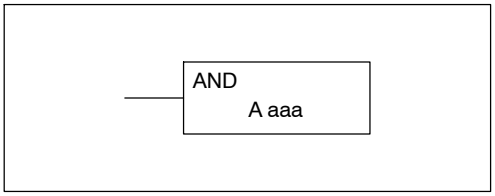
Handheld Programmer Keystrokes

$ 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)

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.



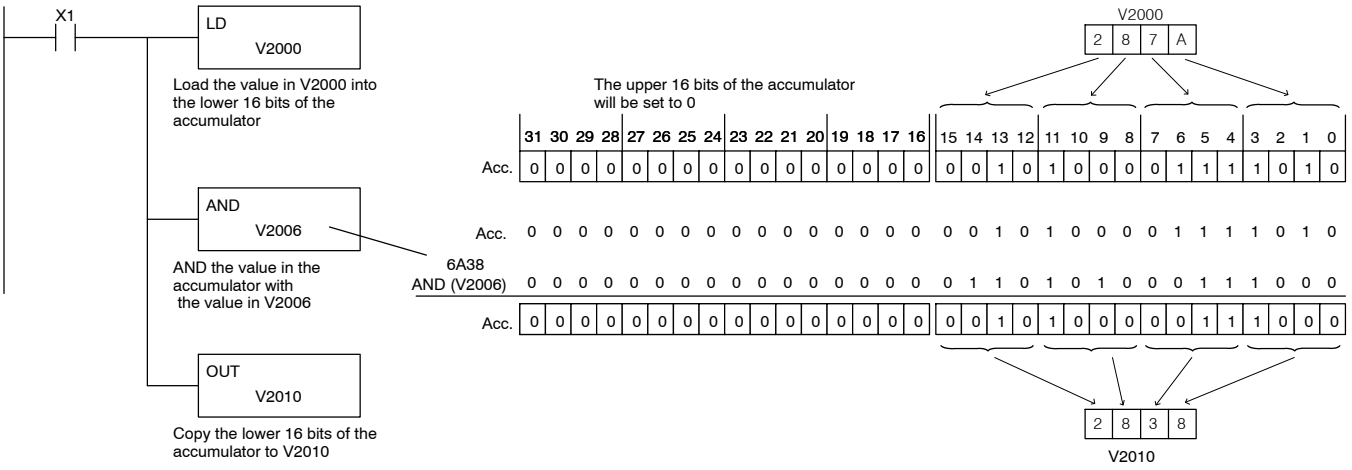
Operand Data Type		DL130 Range
	A	aaa
V-memory	V	All (See page 4-29)

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

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

***Direct*SOFT**

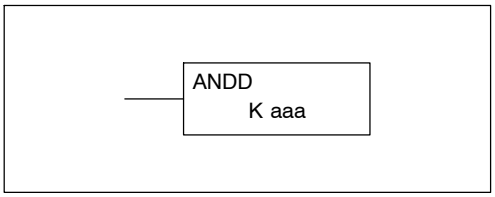


Handheld Programmer Keystrokes

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

And Double (ANDD)

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

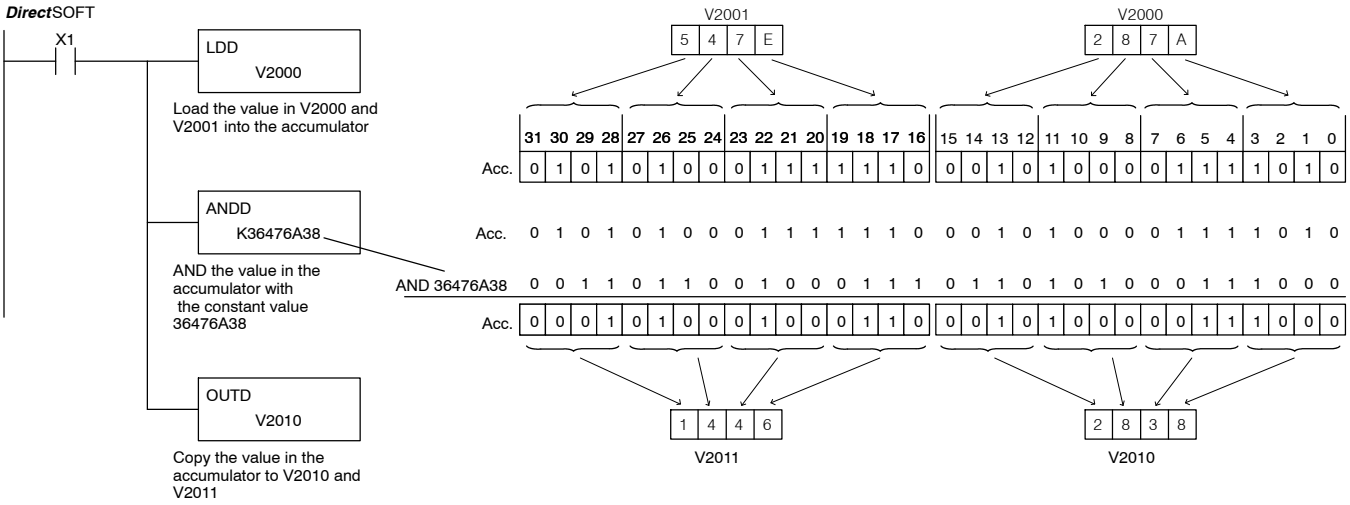


Operand Data Type		DL130 Range
		aaa
V-memory	V	All (See page 4-29)
Constant	K	0-FFFFFFFF

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is the result in the accumulator is negative

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is anded with 36476A38 using the And double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.

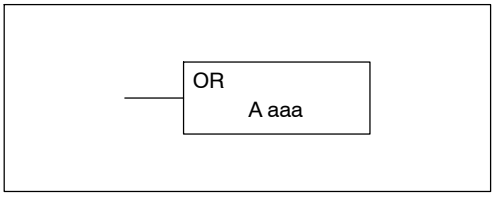


Handheld Programmer Keystrokes

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

Or (OR)

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



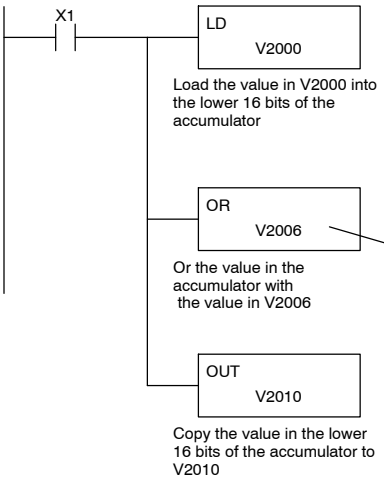
Operand Data Type		DL130 Range
	A	aaa
V-memory	V	All (See page 4-29)

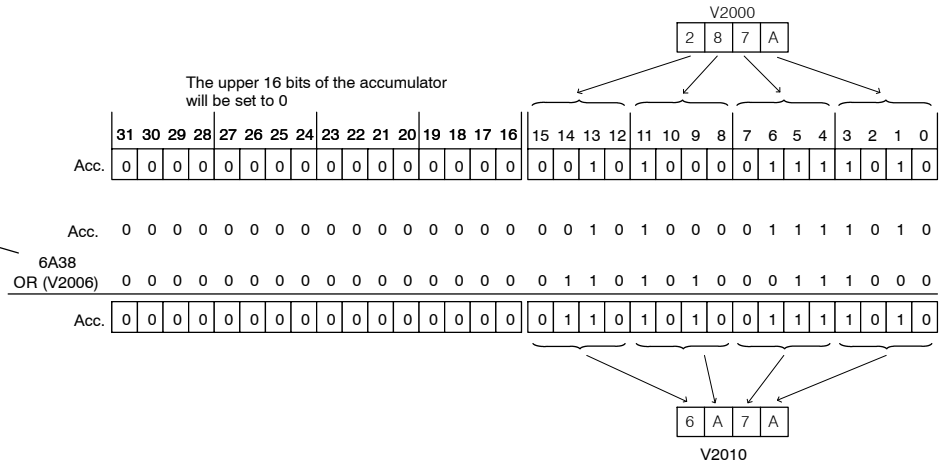
Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

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

***Direct*SOFT**



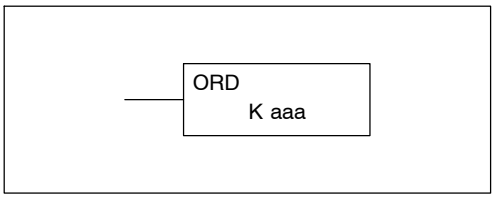


Handheld Programmer Keystrokes

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

Or Double (ORD)

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



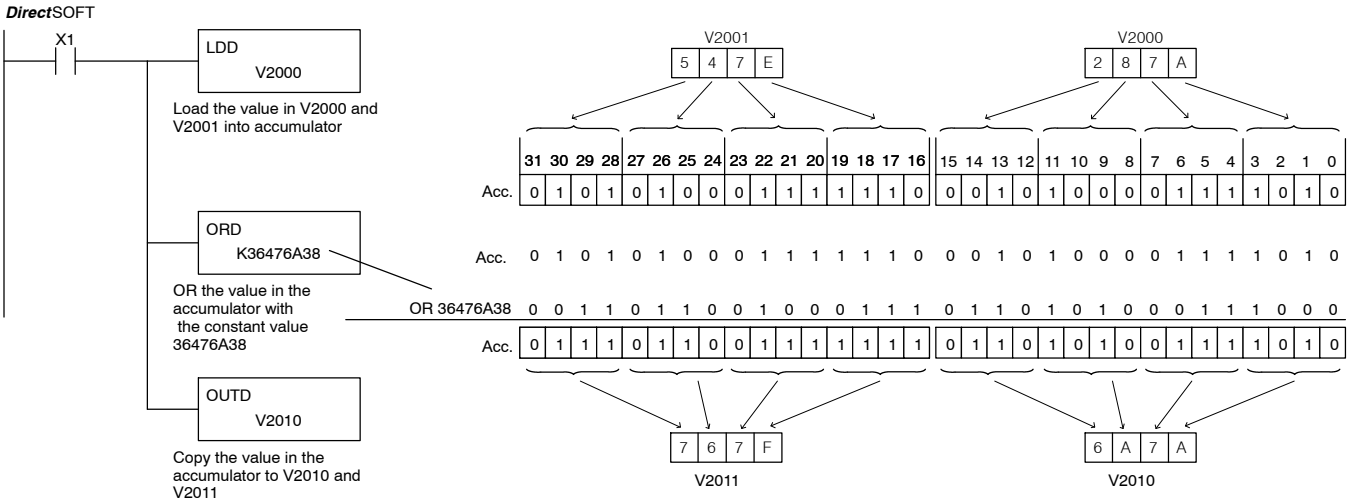
Operand Data Type		DL130 Range
		aaa
V-memory	V	All (See page 4-29)
Constant	K	0-FFFFFFFF

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is the result in the accumulator is negative



NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is ored with 36476A38 using the Or Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.

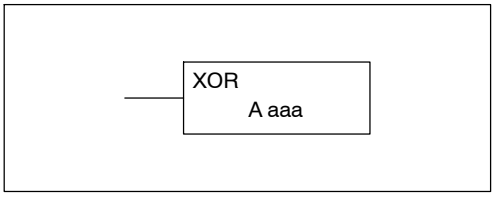


Handheld Programmer Keystrokes

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

Exclusive Or (XOR)

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



Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)

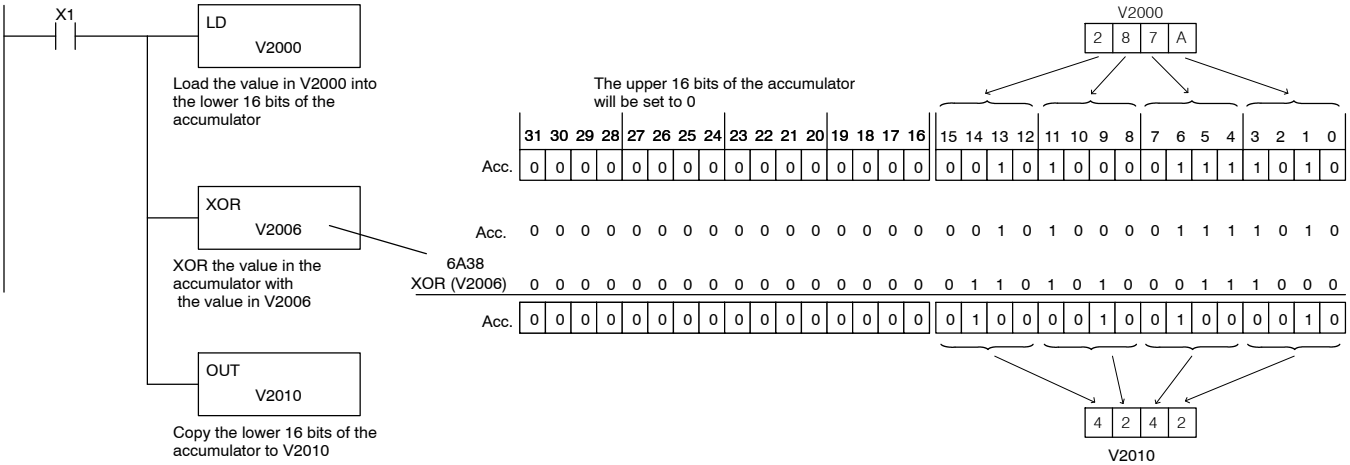
Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero



NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

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

***Direct*SOFT**

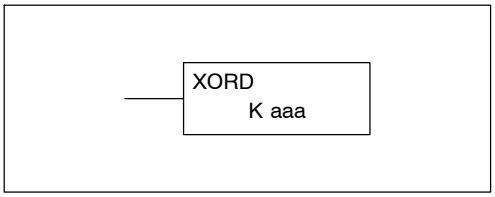


Handheld Programmer Keystrokes

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

Exclusive Or Double (XORD)

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).

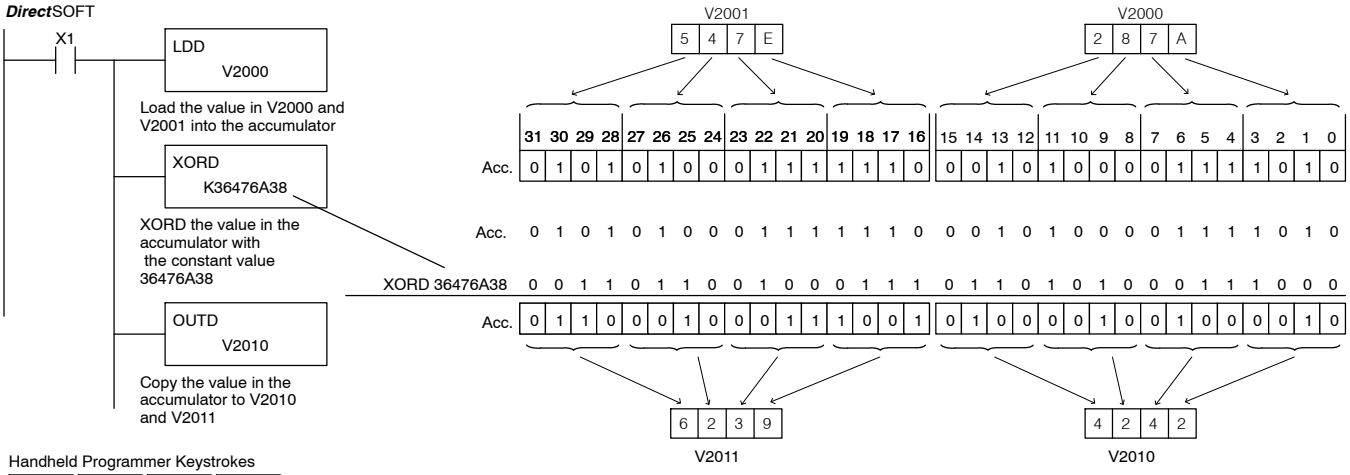


Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)
Constant	K	0-FFFFFFFF

Discrete Bit Flags	Description
SP63	Will be on if the result in the accumulator is zero
SP70	Will be on is the result in the accumulator is negative

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is exclusively ored with 36476A38 using the Exclusive Or Double instruction. The value in the accumulator is output to V2010 and V2011 using the Out Double instruction.

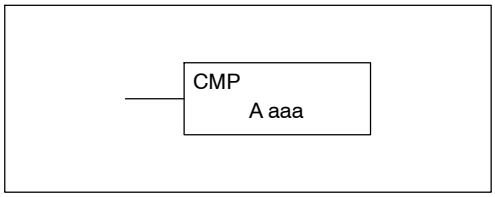


Handheld Programmer Keystrokes

$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	X SET	Q OR	SHFT	D 3	→	SHFT	K JMP		
D 3	G 6	E 4	H 7	G 6	SHFT	A 0	SHFT	D 3	I 8	ENT
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT		

Compare (CMP)

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



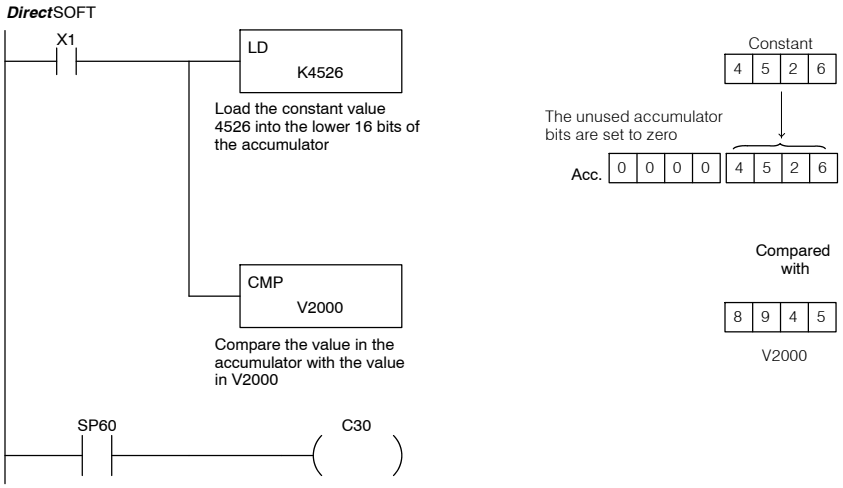
Operand Data Type		DL130 Range
	A	aaa
V-memory	V	All (See page 4-29)

Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.



NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example when X1 is on, the constant 4526 will be loaded into the lower 16 bits of the accumulator using the Load instruction. The value in the accumulator is compared with the value in V2000 using the Compare instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on energizing C30.

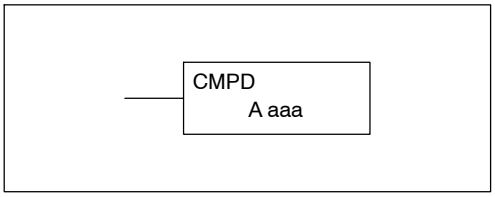


Handheld Programmer Keystrokes

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

Compare Double (CMPD)

The Compare Double instruction is a 32-bit instruction that compares the value in the accumulator with the value (Aaaa), which is either two consecutive V-memory locations or an 8-digit (max.) constant. The corresponding status flag will be turned on indicating the result of the comparison.



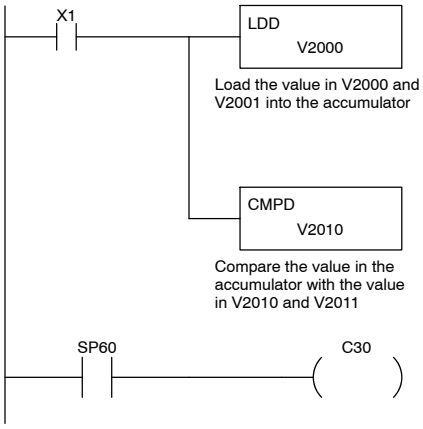
Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)
Constant	K	1-FFFFFFFF

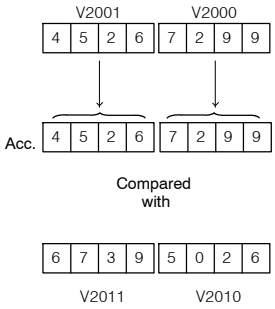
Discrete Bit Flags	Description
SP60	On when the value in the accumulator is less than the instruction value.
SP61	On when the value in the accumulator is equal to the instruction value.
SP62	On when the value in the accumulator is greater than the instruction value.

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in the accumulator is compared with the value in V2010 and V2011 using the CMPD instruction. The corresponding discrete status flag will be turned on indicating the result of the comparison. In this example, if the value in the accumulator is less than the value specified in the Compare instruction, SP60 will turn on energizing C30.

***Direct*SOFT**





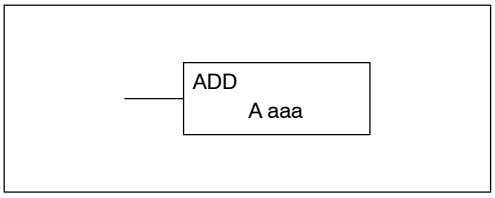
Handheld Programmer Keystrokes

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

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.



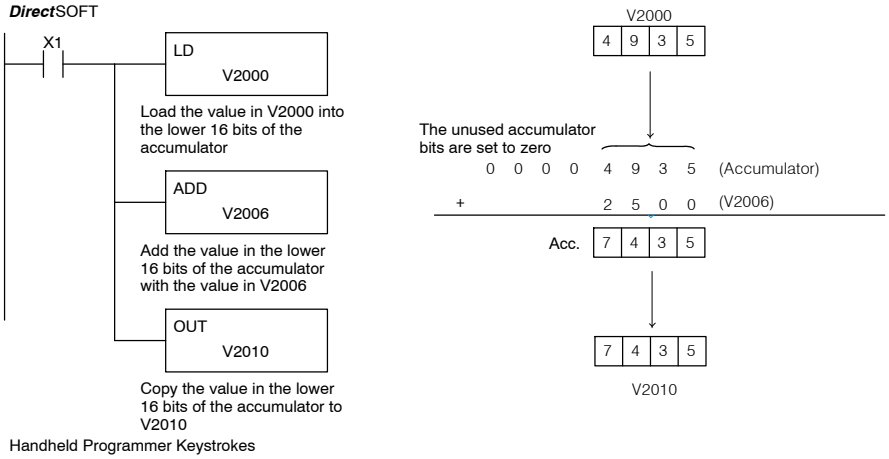
Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)

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



NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in the lower 16 bits of the accumulator are added to the value in V2006 using the Add instruction. The value in the accumulator is copied to V2010 using the Out instruction.

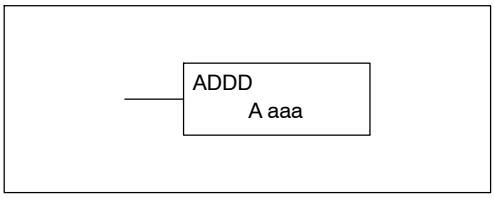


Handheld Programmer Keystrokes

$ 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)

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.



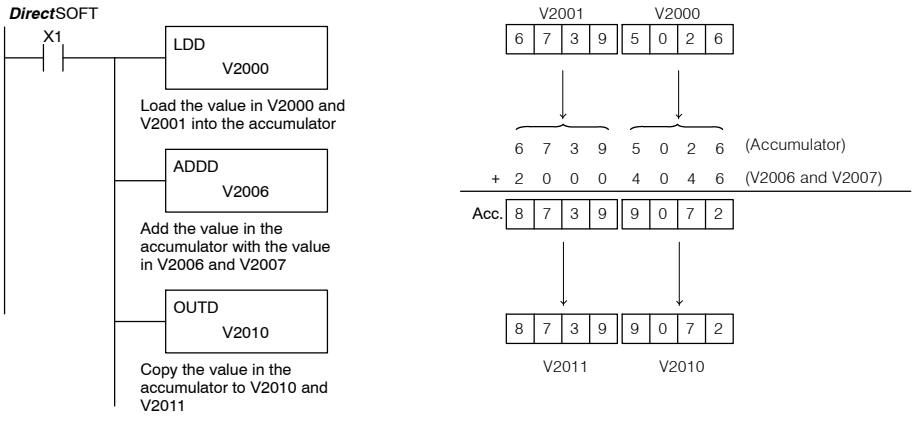
Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)
Constant	K	0-99999999

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



NOTE: 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.

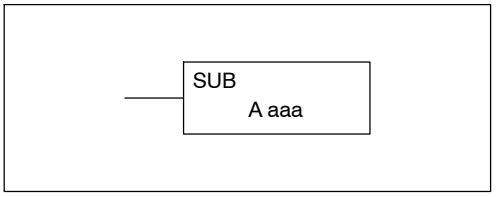


Handheld Programmer Keystrokes

$ 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)

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



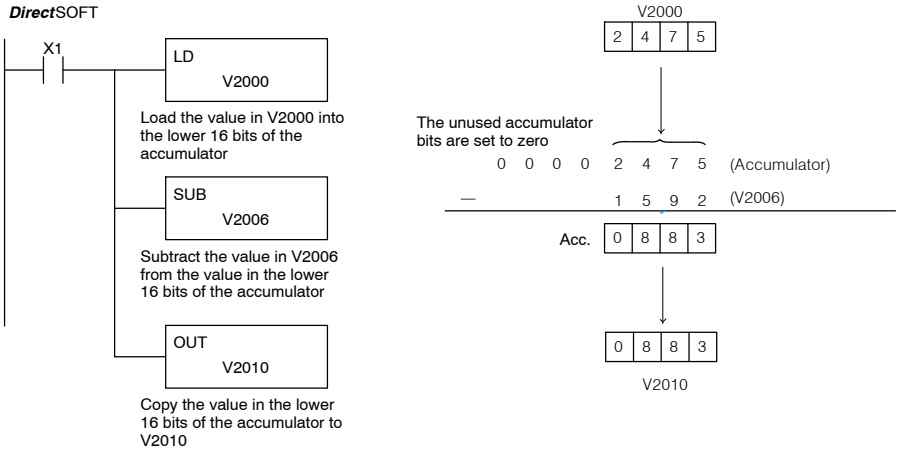
Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)

Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16 bit subtraction instruction results in a borrow.
SP65	On when the 32 bit subtraction instruction results in a borrow.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

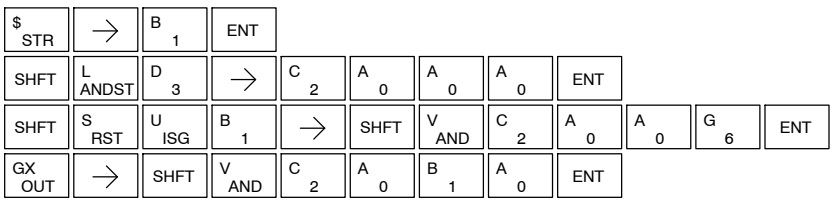


NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, the value in V2000 will be loaded into the accumulator using the Load instruction. The value in V2006 is subtracted from the value in the accumulator using the Subtract instruction. The value in the accumulator is copied to V2010 using the Out instruction.

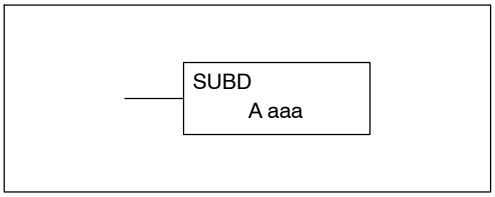


Handheld Programmer Keystrokes



Subtract Double (SUBD)

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



Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)
Constant	K	0-99999999

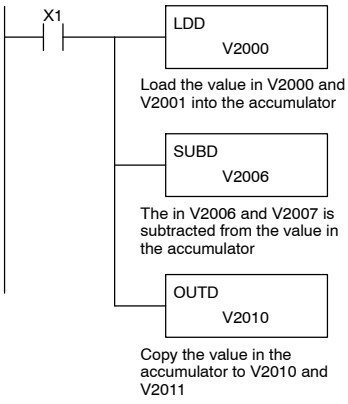
Discrete Bit Flags	Description
SP63	On when the result of the instruction causes the value in the accumulator to be zero.
SP64	On when the 16 bit subtraction instruction results in a borrow.
SP65	On when the 32 bit subtraction instruction results in a borrow.
SP70	On anytime the value in the accumulator is negative.
SP75	On when a BCD instruction is executed and a NON-BCD number was encountered.

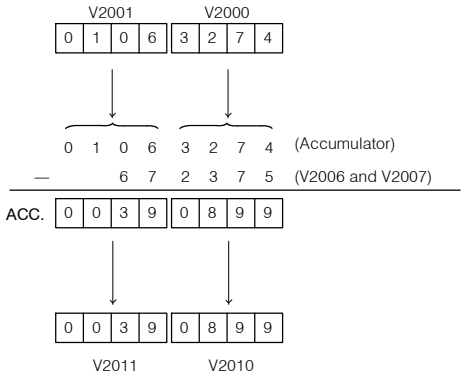


NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

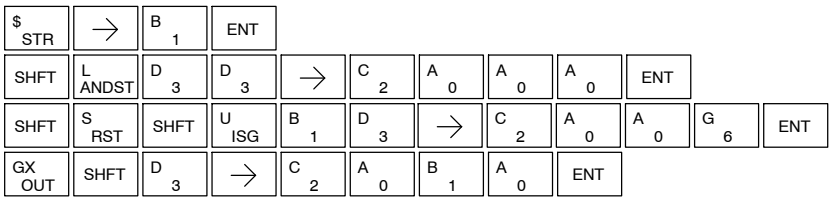
In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The value in V2006 and V2007 is subtracted from the value in the accumulator. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.

***Direct*SOFT**



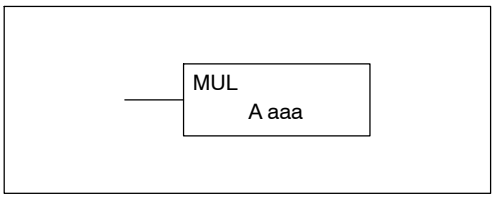


Handheld Programmer Keystrokes



Multiply (MUL)

Multiply is a 16 bit instruction that multiplies the BCD value (Aaaa), which is either a V-memory location or a 4-digit (max.) constant, by the BCD value in the lower 16 bits of the accumulator The result can be up to 8 digits and resides in the accumulator.

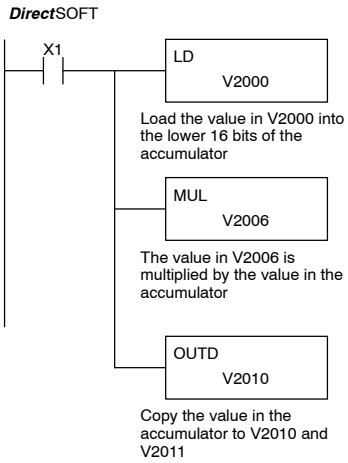


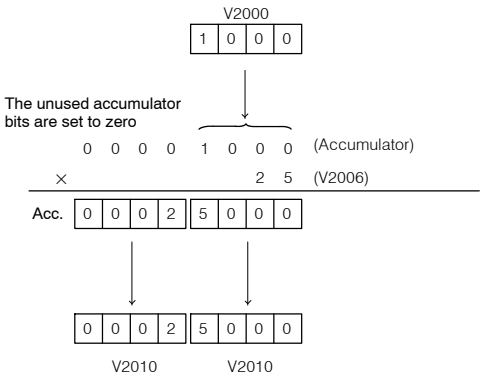
Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)
Constant	K	0-9999

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.



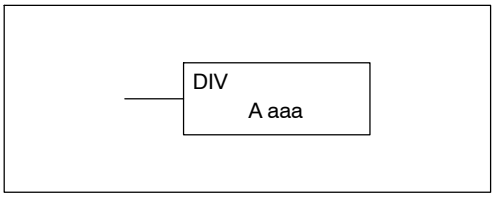


Handheld Programmer Keystrokes

$ 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

Divide (DIV)

Divide is a 16 bit instruction that divides the BCD value in the accumulator by a BCD value (Aaaa), which is either a V-memory location or a 4-digit (max.) constant. The first part of the quotient resides in the accumulator and the remainder resides in the first stack location.

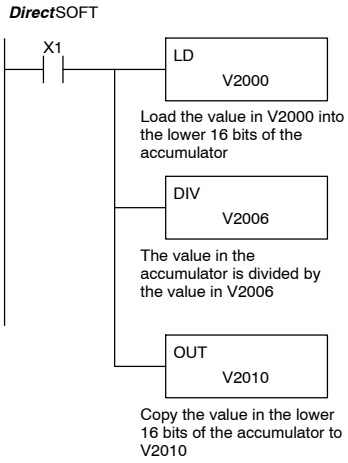


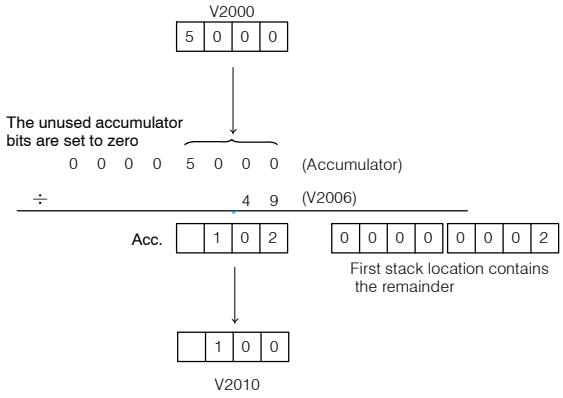
Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)
Constant	K	0-9999

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.



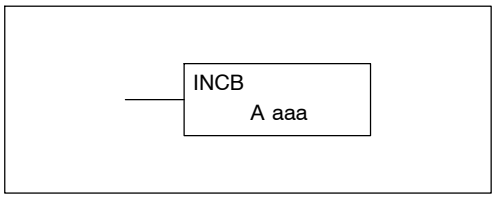


Handheld Programmer Keystrokes

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

Increment Binary (INCB)

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



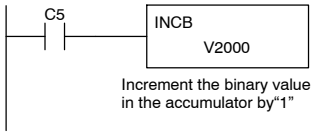
Operand Data Type		DL105 Range
	A	aaa
V-memory	V	All (See page 4-29)

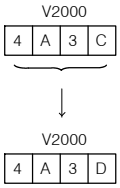
Discrete Bit Flags	Description
SP63	on when the result of the instruction causes the value in the accumulator to be zero.

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

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

***Direct*SOFT**



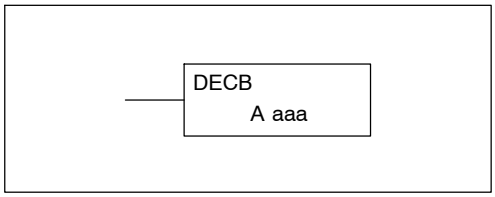


Handheld Programmer Keystrokes

$ STR	→	SHFT	C 2	F 5	ENT							
SHFT	I 8	N TMR	C 2	B 1	→	C 2	A 0	A 0	A 0	ENT		

Decrement Binary (DECB)

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



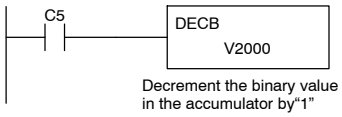
Operand Data Type		DL130 Range
	A	aaa
V-memory	V	All (See page 4-29)

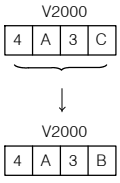
Discrete Bit Flags	Description
SP63	on when the result of the instruction causes the value in the accumulator to be zero.

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.

***Direct*SOFT**





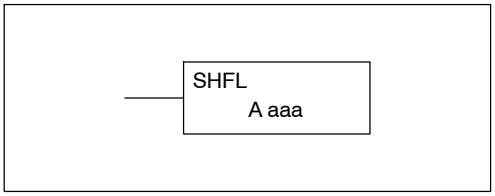
Handheld Programmer Keystrokes

$ STR	→	SHFT	C 2	F 5	ENT							
SHFT	D 3	E 4	C 2	B 1	→	C 2	A 0	A 0	A 0	ENT		

Bit Operation Instructions

Shift Left (SHFL)

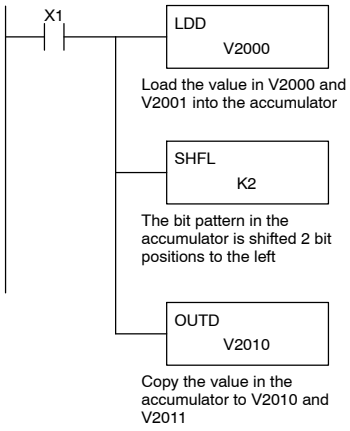
Shift Left is a 32 bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the left. The vacant positions are filled with zeros and the bits shifted out of the accumulator are discarded.

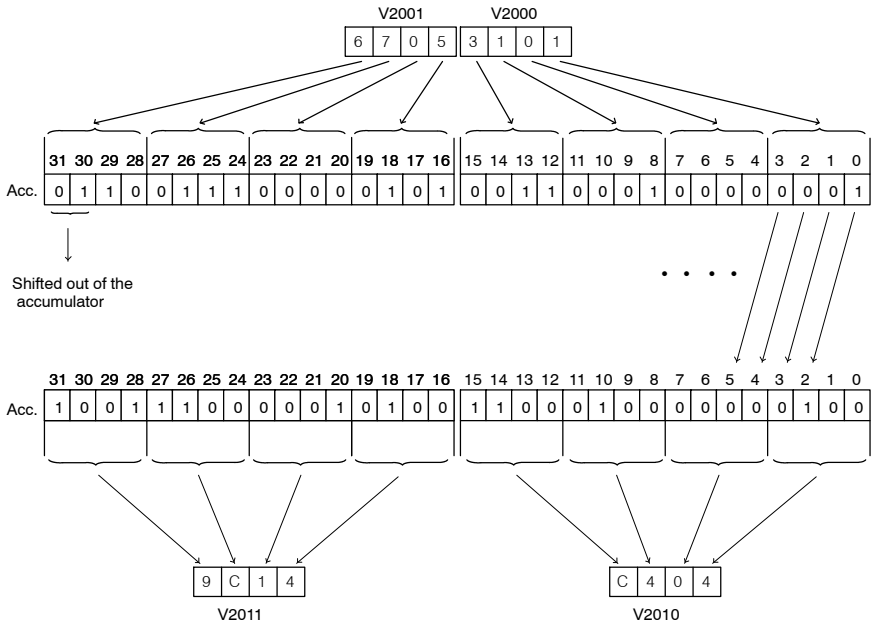


Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)
Constant	K	1-32

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



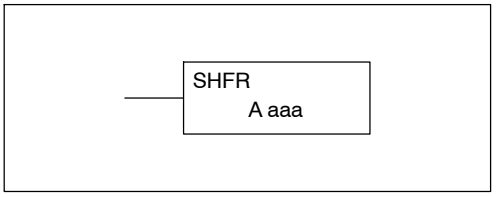


Handheld Programmer Keystrokes

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

Shift Right (SHFR)

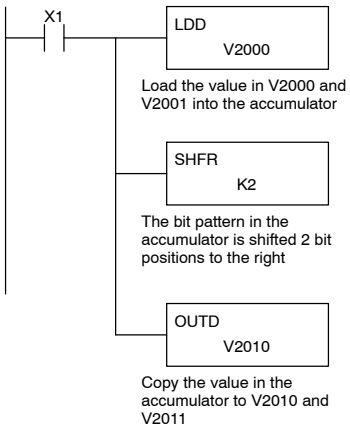
Shift Right is a 32 bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the right. The vacant positions are filled with zeros and the bits shifted out of the accumulator are lost.

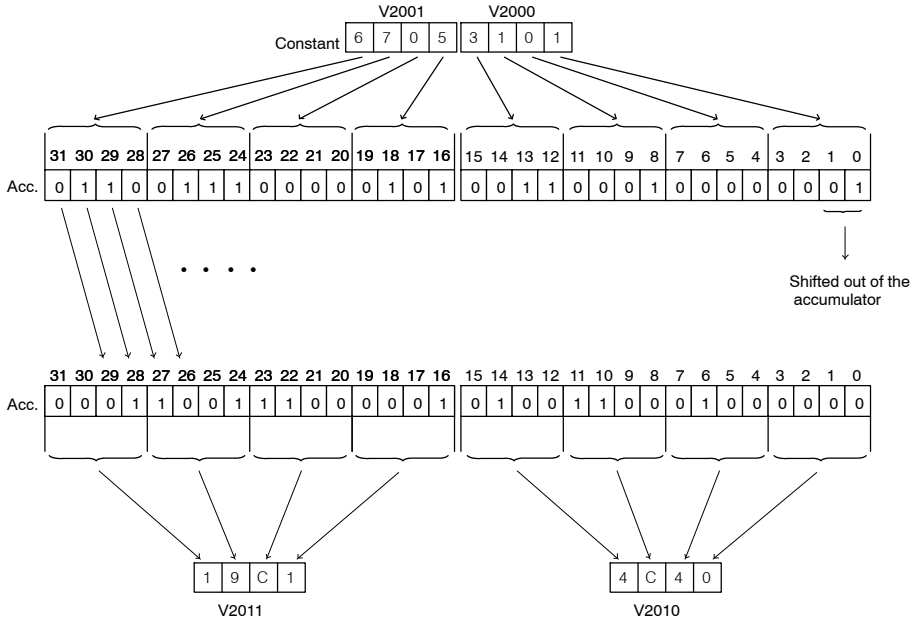


Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-29)
Constant	K	1-32

In the following example, when X1 is on, the value in V2000 and V2001 will be loaded into the accumulator using the Load Double instruction. The bit pattern in the accumulator is shifted 2 bits to the right using the Shift Right instruction. The value in the accumulator is copied to V2010 and V2011 using the Out Double instruction.

***Direct*SOFT**



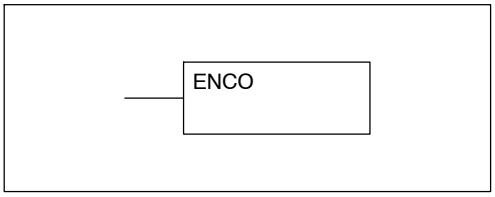


Handheld Programmer Keystrokes

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

Encode (ENCO)

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

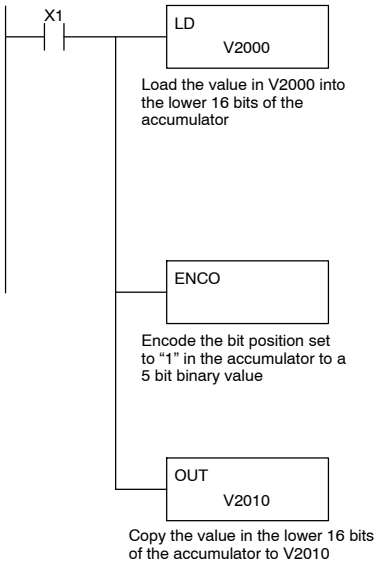


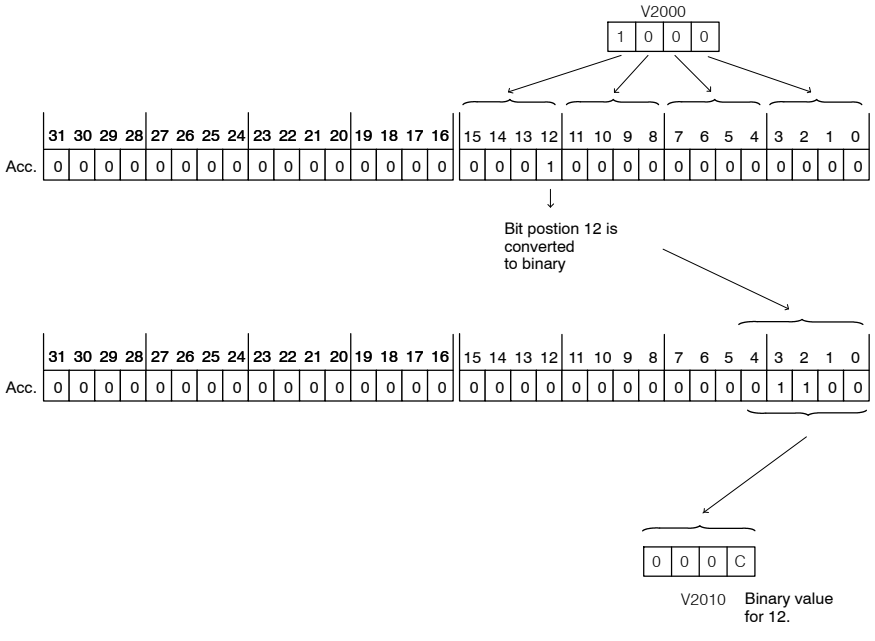
Discrete Bit Flags	Description
SP53	On when the value of the operand is larger than the accumulator can work with.

NOTE: The status flags are only valid until another instruction that uses the same flags is executed.

In the following example, when X1 is on, The value in V2000 is loaded into the accumulator using the Load instruction. The bit position set to a "1" in the accumulator is encoded to the corresponding 5 bit binary value using the Encode instruction. The value in the lower 16 bits of the accumulator is copied to V2010 using the Out instruction.

***Direct*SOFT**



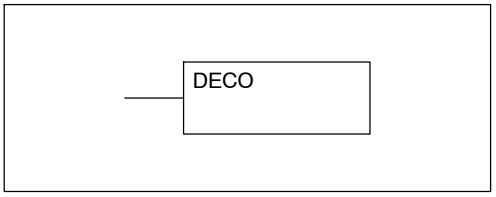


Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	L ANDST	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	E 4	N TMR	C 2	O INST#	ENT			
GX OUT	→	SHFT	V AND	C 2	A 0	B 1	A 0	ENT

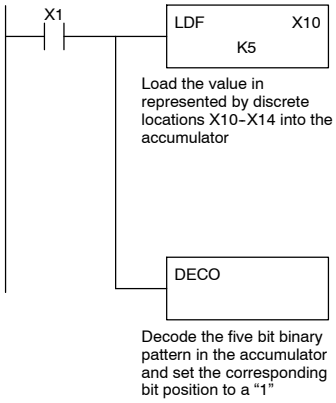
Decode (DECO)

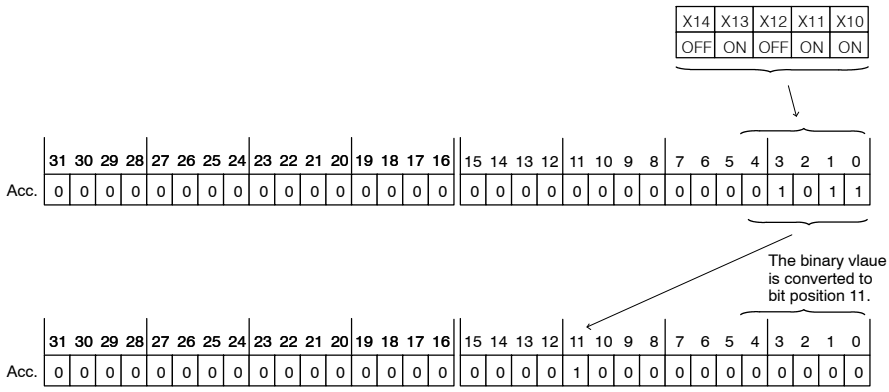
The Decode instruction decodes a 5 bit binary value of 0–31 (0–1F HEX) in the accumulator by setting the appropriate bit position to a 1. If the accumulator contains the value F (HEX), bit 15 will be set in the accumulator. If the value to be decoded is greater than 31, the number is divided by 32 until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



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.

***Direct*SOFT**





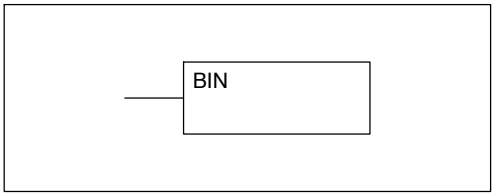
Handheld Programmer Keystrokes

$ 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)

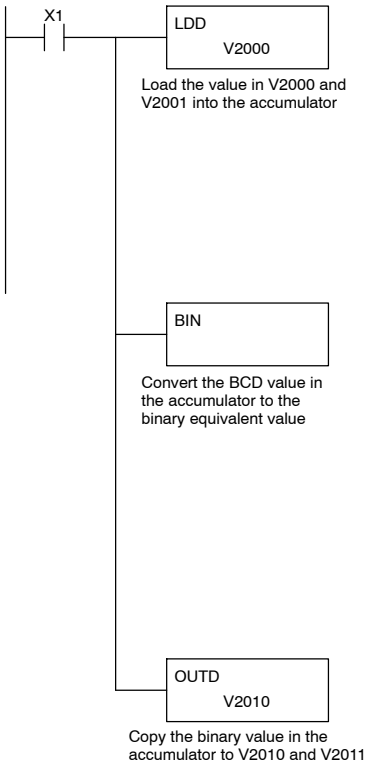
Binary (BIN)

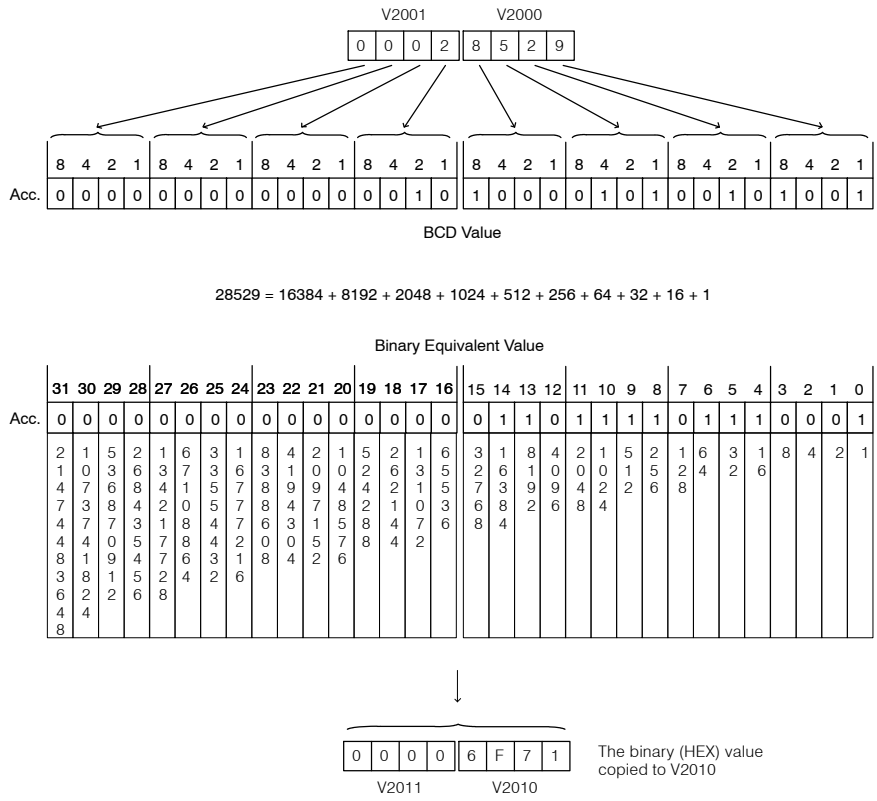
The Binary instruction converts a BCD value in the accumulator to the equivalent binary value. The result resides in the accumulator.



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.)

DirectSOFT



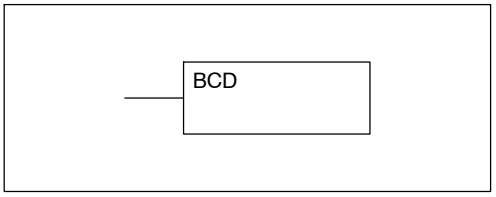


Handheld Programmer Keystrokes

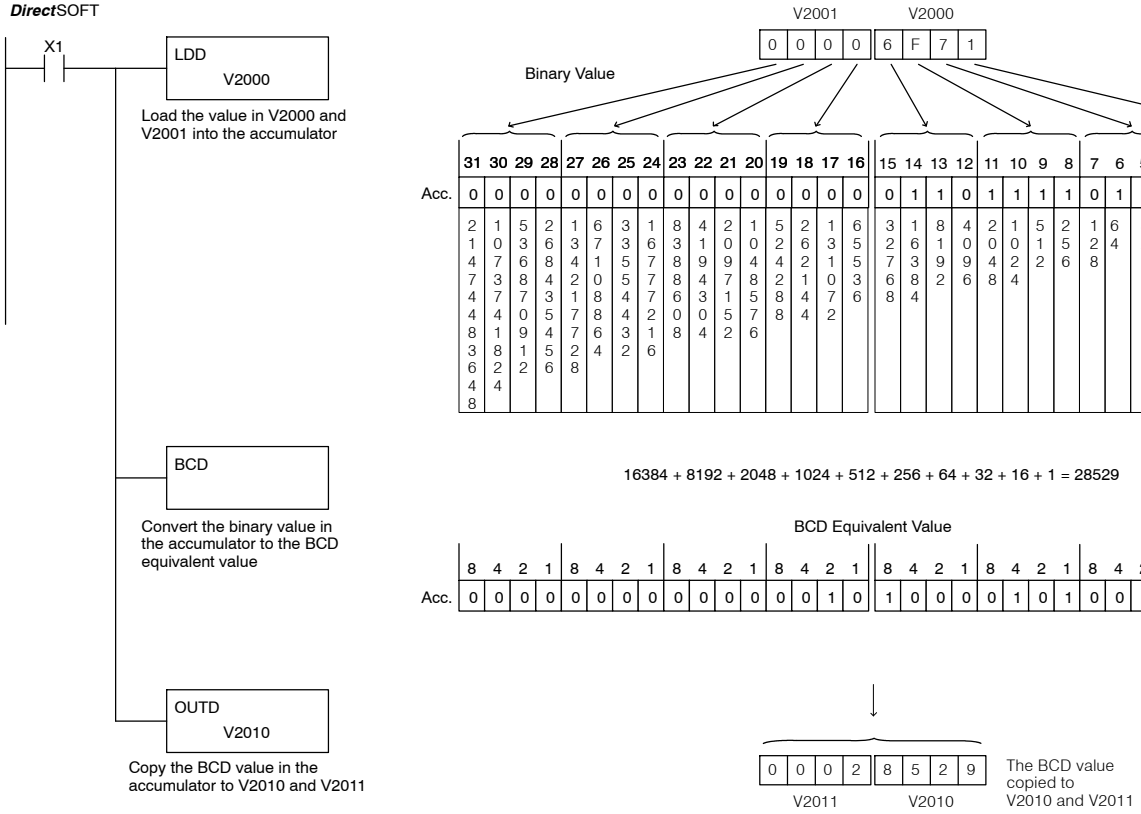
$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	B 1	I 8	N TMR	ENT					
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

Binary Coded Decimal (BCD)

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



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

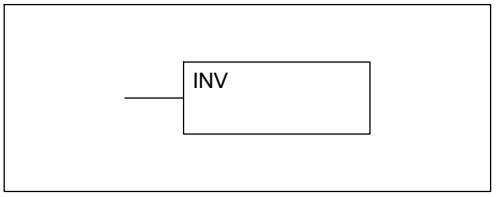


Handheld Programmer Keystrokes

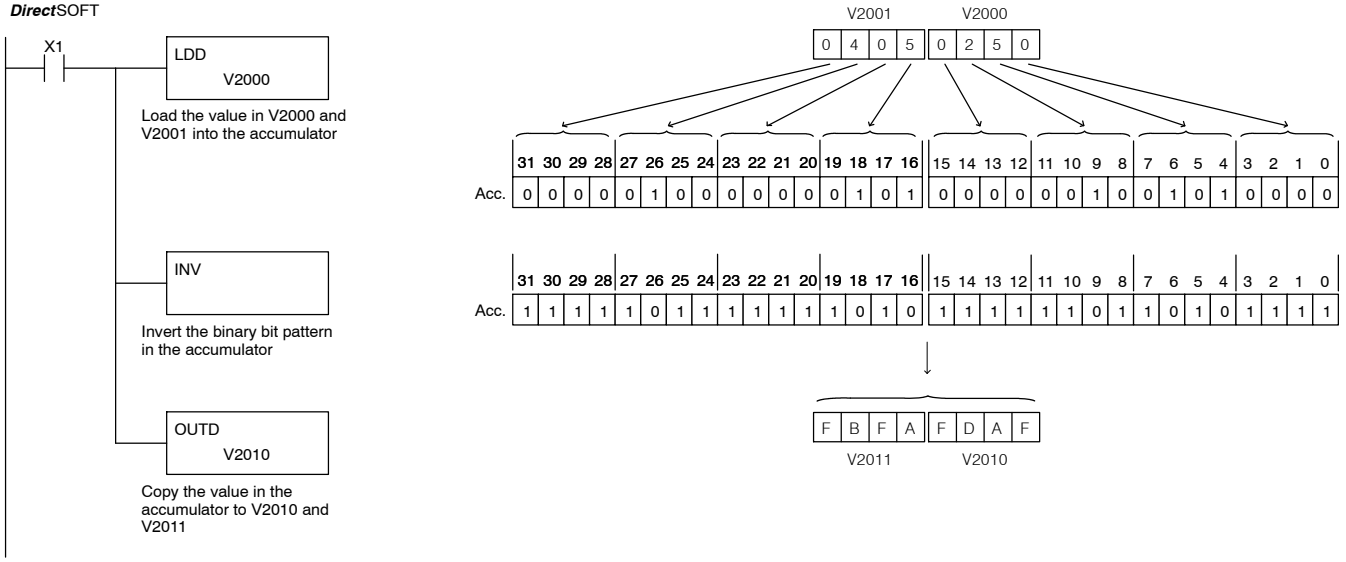
$ STR	→	B 1	ENT						
SHFT	L ANDST	D 3	D 3	→	C 2	A 0	A 0	A 0	ENT
SHFT	B 1	C 2	D 3	ENT					
GX OUT	SHFT	D 3	→	C 2	A 0	B 1	A 0	ENT	

Invert (INV)

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



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



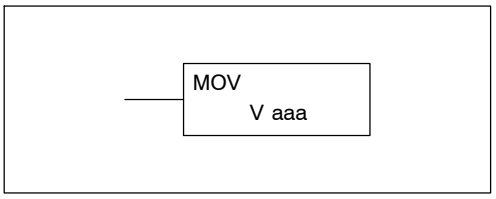
Handheld Programmer Keystrokes

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

Table Instructions

Move (MOV)

The Move instruction moves the values from a V-memory table to another V-memory table the same length. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Move function.



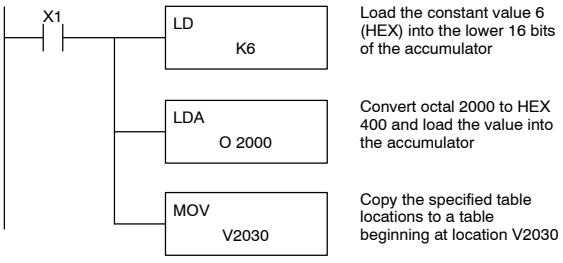
- Step 1:— Load the number of V-memory locations to be moved into the first level of the accumulator stack. This parameter is a HEX value (K40 max, 100 octal).
- Step 2:— Load the starting V-memory location for the locations to be moved into the accumulator. This parameter is a HEX value.
- Step 3:— Insert the MOVE instruction which specifies starting V-memory location (Vaaa) for the destination table.

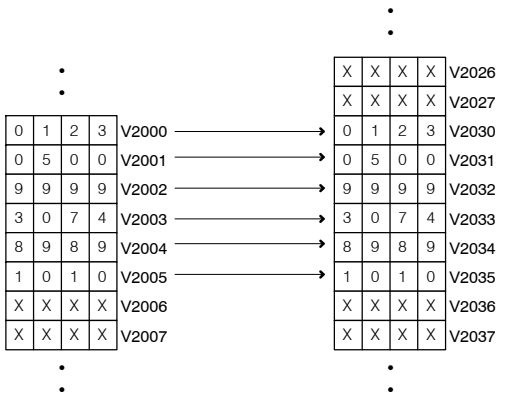
Helpful Hint: — For parameters that require HEX values when referencing memory locations, the LDA instruction can be used to convert an octal address to the HEX equivalent and load the value into the accumulator.

Operand Data Type	DL130 Range
	aaa
V-memory V	All (See page 4-28)

In the following example, when X1 is on, the constant value (K6) is loaded into the accumulator using the Load instruction. This value specifies the length of the table and is placed in the first stack location after the Load Address instruction is executed. The octal address 2000 (V2000), the starting location for the source table is loaded into the accumulator. The destination table location (V2030) is specified in the Move instruction.

***Direct*SOFT**





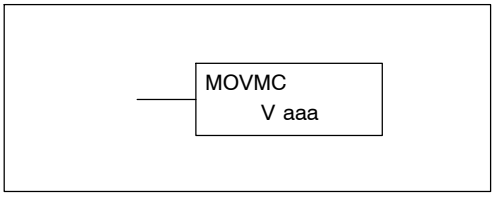
Handheld Programmer Keystrokes

$ 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

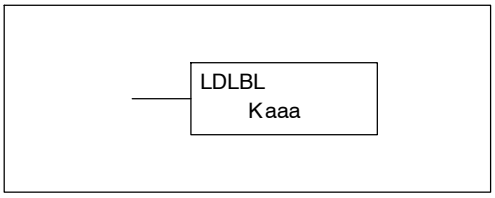
Move Memory Cartridge / Load Label

(MOVMC), (LDLBL)

The Move Memory Cartridge instruction is used to copy data between V memory and program ladder memory. The Load Label instruction is *only* used with the MOVMC instruction when copying data *from* program ladder memory *to* V-memory.



To copy data between V-memory and program ladder memory, the function parameters are loaded into the first two levels of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the 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. Load the *source address* (LDA Oaaa) into the accumulator when copying data from V-memory to ladder memory. This is the source location of the value. Source addresses in V-memory must be entered in HEX.
- Step 4:— Insert the MOVMC instruction which specifies destination V-memory (Vaaa), or data label (Kaaa). This is the copy destination.

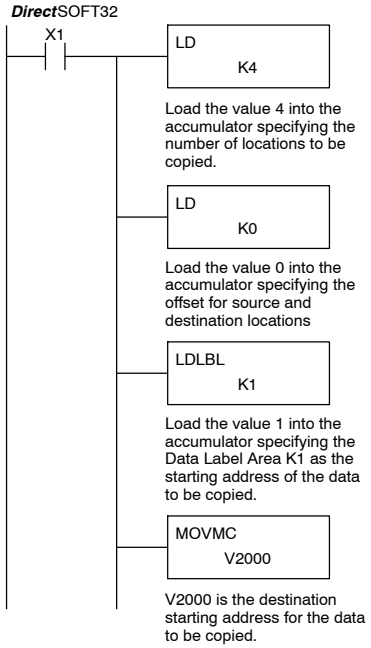
Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-28)

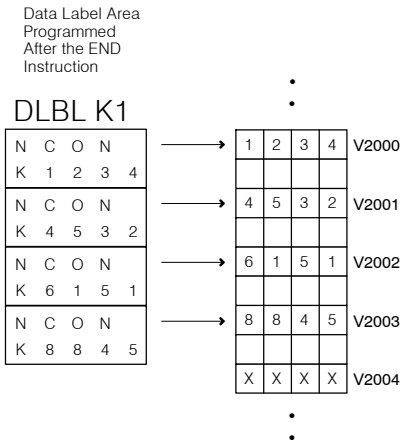


NOTE: See Appendix E for an explanation of the DL105 memory system.

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.





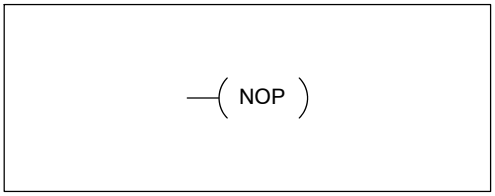
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT								
SHFT	L ANDST	D 3	→	SHFT	K JMP	E 4	ENT				
SHFT	L ANDST	D 3	→	SHFT	K JMP	A 0	ENT				
SHFT	L ANDST	D 3	L ANDST	B 1	L ANDST	→	B 1	ENT			
SHFT	M ORST	O INST#	V AND	M ORST	C 2	→	C 2	A 0	A 0	A 0	ENT

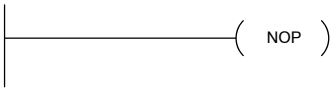
CPU Control Instructions

No Operation (NOP)

The No Operation is an empty (not programmed) memory location.



DirectSOFT

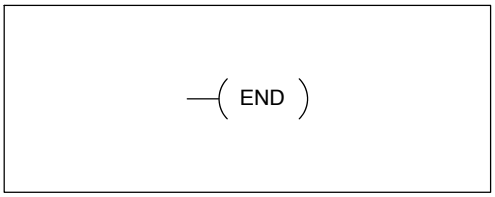


Handheld Programmer Keystrokes

SHFT	N TMR	O INST#	P CV	ENT

End (END)

The End instruction marks the termination point of the normal program scan. An End instruction is required at the end of the main program body. If the End instruction is omitted an error will occur and the CPU will not enter the Run Mode. Data labels, subroutines and interrupt routines are placed after the End instruction. The End instruction is not conditional; therefore, no input contact is allowed.



DirectSOFT

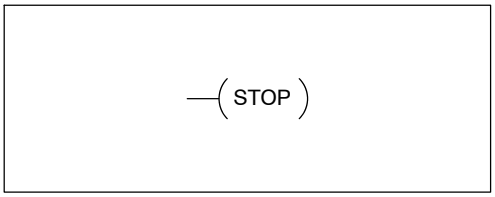


Handheld Programmer Keystrokes

SHFT	E 4	N TMR	D 3	ENT

Stop (STOP)

The Stop instruction changes the operational mode of the CPU from Run to Program (Stop) mode. This instruction is typically used to stop PLC operation in an error condition.



In the following example, when C0 turns on, the CPU will stop operation and switch to the program mode.

DirectSOFT



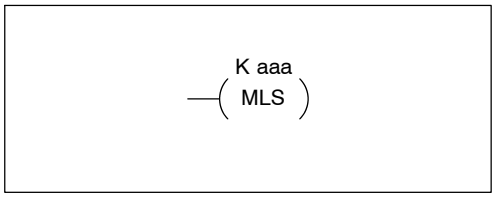
Handheld Programmer Keystrokes

$ STR	→	SHFT	C 2	A 0	ENT	
SHFT	S RST	SHFT	T MLR	O INST#	P CV	ENT

Program Control Instructions

Master Line Set (MLS)

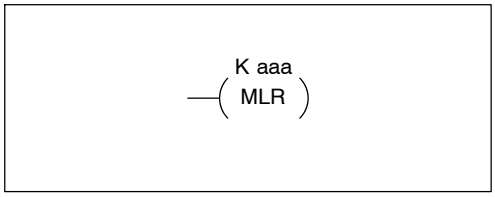
The Master Line Set instruction allows the program to control sections of ladder logic by forming a new power rail controlled by the main left power rail. The main left rail is always master line 0. When a MLS K1 instruction is used, a new power rail is created at level 1. Master Line Sets and Master Line Resets can be used to nest power rails up to seven levels deep.



Operand Data Type		DL130 Range
		aaa
Constant	K	1-7

Master Line Reset (MLR)

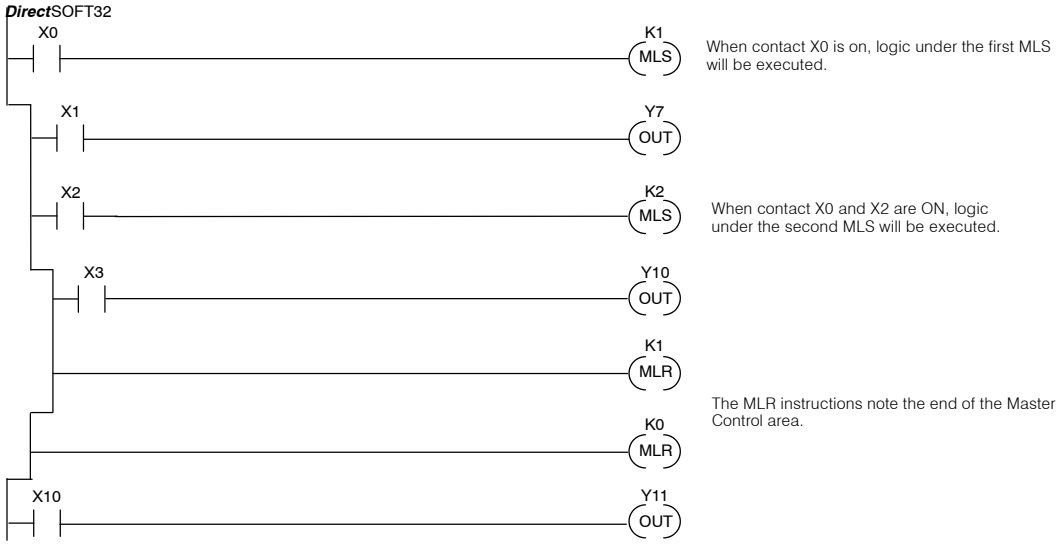
The Master Line Reset instruction marks the end of control for the corresponding MLS instruction. The MLR reference is one less than the corresponding MLS.



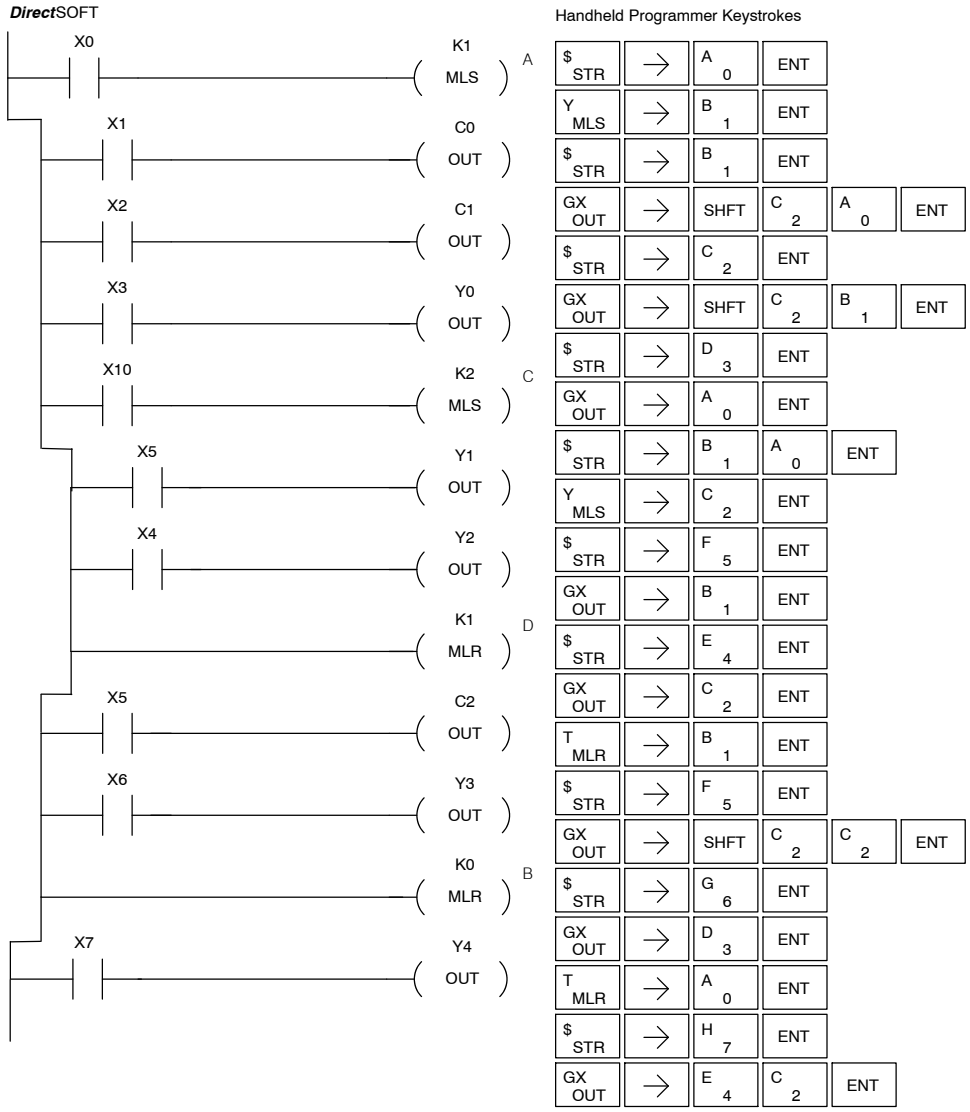
Operand Data Type		DL130 Range
		aaa
Constant	K	0-6

Understanding Master Control Relays

The Master Line Set (MLS) and Master Line Reset (MLR) instructions allow you to quickly enable (or disable) sections of the RLL program. This provides program control flexibility. The following example shows how the MLS and MLR instructions operate by creating a sub power rail for control logic.



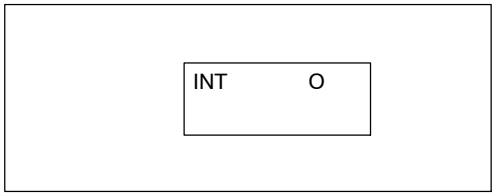
MLS/MLR Example In the following MLS/MLR example logic between the first MLS K1 (A) and MLR K0 (B) will function only if input X0 is on. The logic between the MLS K2 (C) and MLR K1 (D) will function only if input X10 and X0 is on. The last rung is not controlled by either of the MLS coils.



Interrupt Instructions

Interrupt (INT)

The Interrupt instruction allows a section of ladder logic to be placed below the main body of the program and executed only when needed. High-Speed I/O Modes 10, 20, and 40 can generate an interrupt. With Mode 40, you may select an external interrupt (input X0), or a time-based interrupt (5–999 mS).



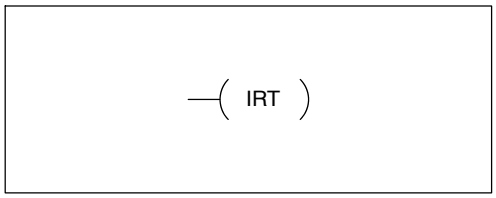
Typically, interrupts are used in an application when a fast response to an input is needed or a program section must execute faster than the normal CPU scan. The interrupt label and all associated logic must be placed after the End statement in the program. When an interrupt occurs, the CPU will complete 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 DL105, only one interrupt is available.

Operand Data Type		DL130 Range
Constant	O	0

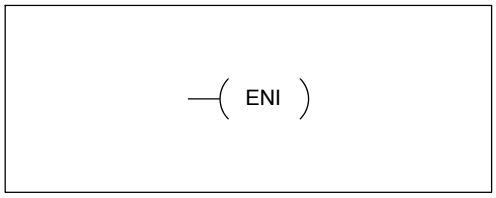
Interrupt Return (IRT)

An Interrupt Return is normally executed as the last instruction in the interrupt routine. It returns the CPU to the point in the main program from which it was called. The Interrupt Return is a stand-alone instruction (no input contact on the rung).



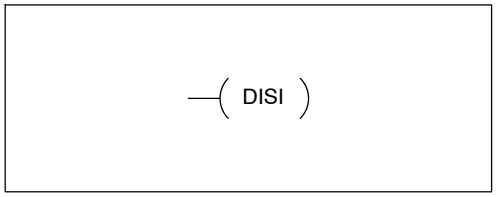
Enable Interrupts (ENI)

The Enable Interrupt instruction is placed in the main ladder program (before the End instruction), enabling the interrupt. The interrupt remains enabled until the program executes a Disable Interrupt instruction.



Disable Interrupts (DISI)

A Disable Interrupt instruction in the main body of the application program (before the End instruction) will disable the interupt (either extenal or timed). The interrupt remains disabled until the program executes an Enable Interrupt instruction.

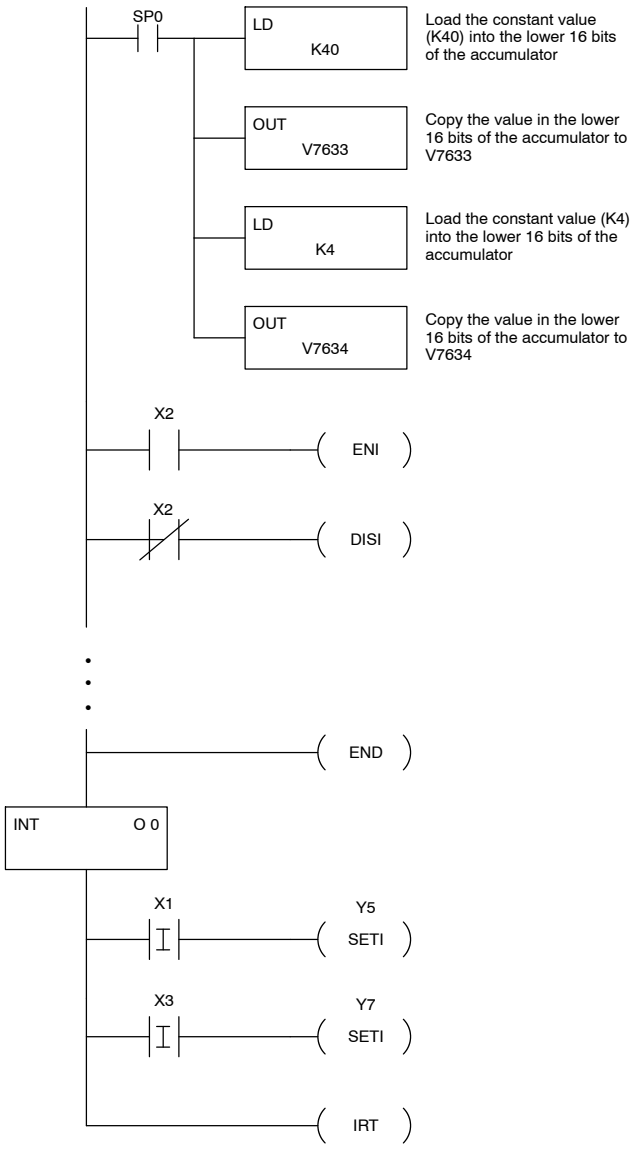


External Interrupt Program Example

In the following example, we do some initialization on the first scan, using the first-scan contact SP0. The interrupt feature is the HSIO Mode 40. Then we configure X0 as the external interrupt by writing to its configuration register, V7634. See Chapter 3, Mode 40 Operation for more details.

During program execution, when X2 is on the interrupt is enabled. When X2 is off the interrupt will be disabled. When an interrupt signal (X0) occurs the CPU will jump to the interrupt label INT O 0. The application ladder logic in the interrupt routine will be performed. The CPU will return to the main body of the program after the IRT instruction is executed.

***Direct*SOFT**



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		

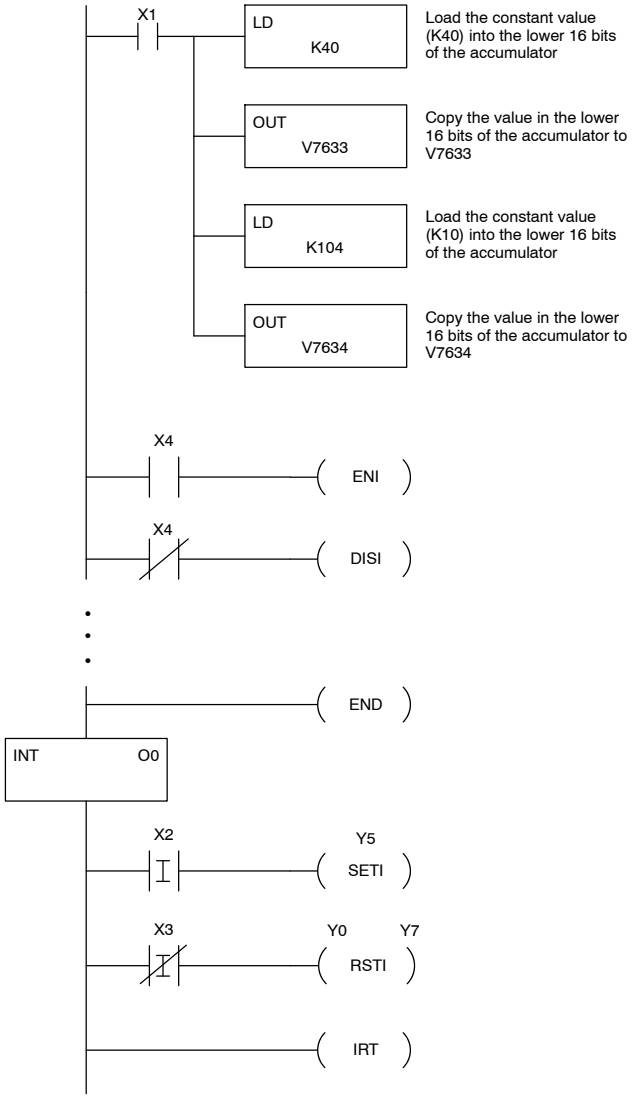
NOTE: Only one interrupt is available in the DL105 and it must be Int 0.

Timed Interrupt Program Example

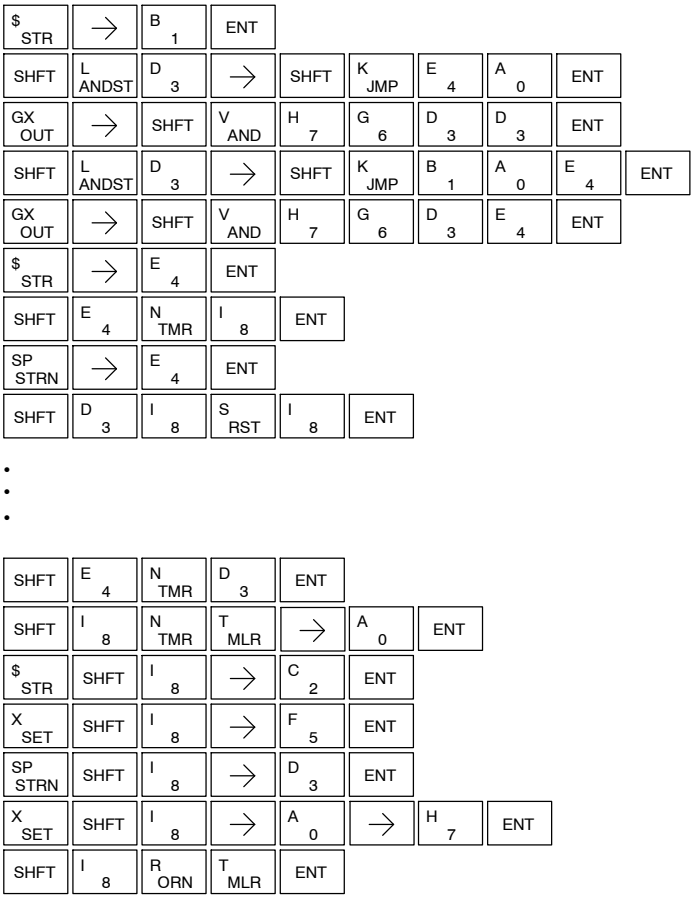
In the following example, we do some initialization on the first scan, using the first-scan contact SP0. The interrupt feature is the HSIO Mode 40. Then we configure the HSIO timer as a 10 mS interrupt by writing K104 to the configuration register for X0 (V7634). See Chapter 3, Mode 40 Operation for 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.

***Direct*SOFT**



Handheld Programmer Keystrokes

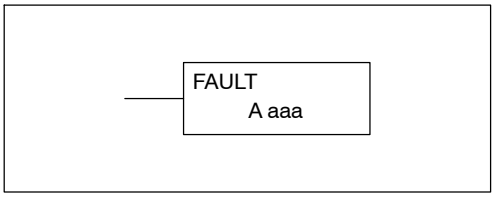


NOTE: Only one interrupt is available in the DL105 and it must be Int 0.

Message Instructions

Fault (FAULT)

The Fault instruction is used to display a message on the handheld programmer or in the ***Direct***SOFT 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.

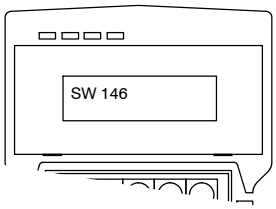
Operand Data Type		DL130 Range
	A	**aaa**
V-memory	V	All (See page 4-28)
Constant	K	1-FFFF

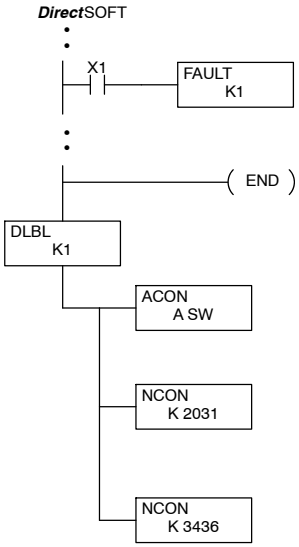


NOTE: The FAULT instruction takes a considerable amount of time to execute. This is because the FAULT parameters are stored in EEPROM. Be sure to consider the instructions execution times (shown in Appendix C) if you are attempting to use the FAULT instructions in applications that require faster than normal execution cycles.

Fault Example

In the following example when X1 is on, the message SW 146 will display on the handheld programmer. The NCONs use the HEX ASCII equivalent of the text to be displayed. (The HEX ASCII for a blank is 20, a 1 is 31, 4 is 34 ...)





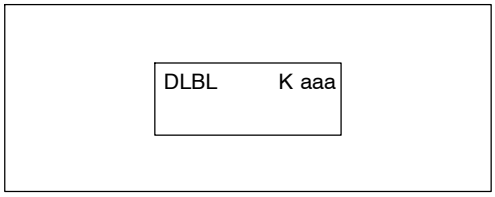
Handheld Programmer Keystrokes

$ STR	→	B 1	ENT					
SHFT	F 5	A 0	U ISG	L ANDST	T MLR	→	B 1	ENT

SHFT	E 4	N TMR	D 3	ENT						
SHFT	D 3	L ANDST	B 1	L ANDST	→	B 1	ENT			
SHFT	A 0	C 2	O INST#	N TMR	→	S RST	W ANDN	ENT		
SHFT	N TMR	C 2	O INST#	N TMR	→	C 2	A 0	D 3	B 1	ENT
SHFT	N TMR	C 2	O INST#	N TMR	→	D 3	E 4	D 3	G 6	ENT

Data Label (DLBL)

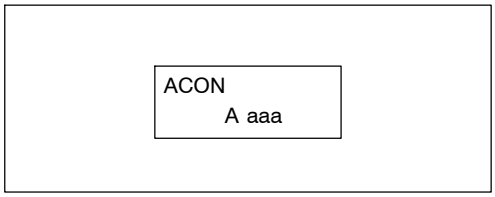
The Data Label instruction marks the beginning of an ASCII / numeric data area. DLBLs are programmed after the End statement. A maximum of 32 DLBL instructions can be used in a program. Multiple NCONs and ACONs can be used in a DLBL area.



Operand Data Type		DL130 Range
		aaa
Constant	K	1-FFFF

ASCII Constant (ACON)

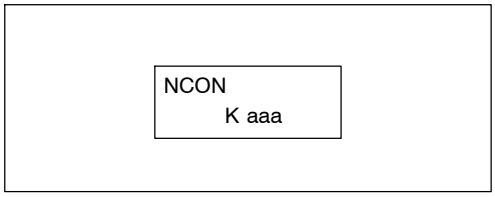
The ASCII Constant instruction is used with the DLBL instruction to store ASCII text for use with other instructions. Two ASCII characters can be stored in an ACON instruction. If only one character is stored in a ACON a leading space will be inserted.



Operand Data Type		DL130 Range
		aaa
ASCII	A	0-9 A-Z

Numerical Constant (NCON)

The Numerical Constant instruction is used with the DLBL instruction to store the HEX ASCII equivalent of numerical data for use with other instructions. Two digits can be stored in an NCON instruction.

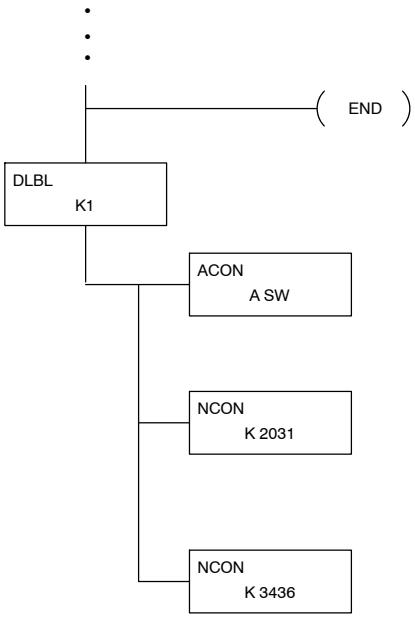


Operand Data Type		DL130 Range
		aaa
Constant	K	0-FFFF

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.

***Direct*SOFT**



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

6

Drum Instruction Programming

In This Chapter. . . .

- — Introduction
- — Step Transitions
- — Overview of Drum Operation
- — Drum Control Techniques
- — Drum Instruction

Introduction

Purpose

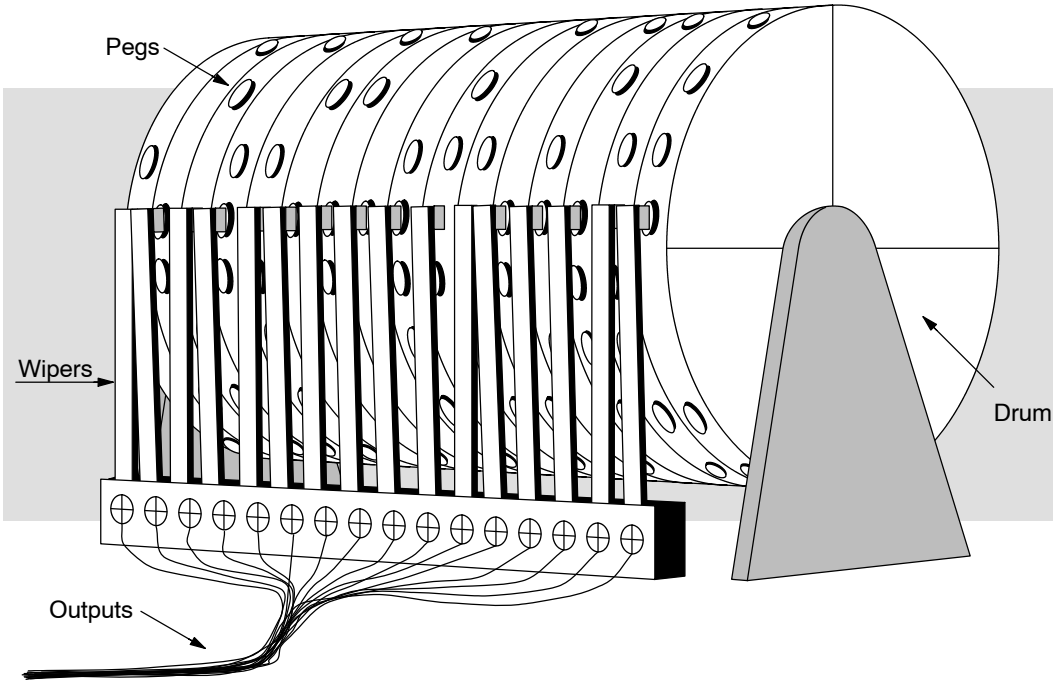
The Event Drum (EDRUM) instruction in the F1-130 CPU electronically simulates an electro-mechanical drum sequencer. The instruction offers enhancements to the basic principle, which we describe first.

Drum Terminology

Drum instructions are best suited for repetitive processes that consist of a finite number of steps. They can do the work of many rungs of ladder logic with elegant simplicity. Therefore, drums can save a lot of programming and debugging time.

We introduce some terminology associated with the drum instruction by describing the original mechanical drum shown below. The mechanical **drum** generally has pegs on its curved surface. The pegs are populated in a particular **pattern**, representing a set of desired actions for machine control. A motor or solenoid rotates the drum a precise amount at specific times. During rotation, stationary wipers sense the presence of pegs (present = on, absent = off). This interaction makes or breaks electrical contact with the wipers, creating electrical **outputs** from the drum. The outputs are wired to devices on a machine for On/Off control.

Drums usually have a finite number of positions within one rotation, called **steps**. Each step represents some process step. At powerup, the drum **resets** to a particular step. The drum rotates from one step to the next based on a **timer**, or on some external **event**. During special conditions, a machine operator can manually increment the drum step using a **jog** control on the drum's drive mechanism. The contact closure of each wiper generates a unique on/off pattern called a **sequence**, designed for controlling a specific machine. Because the drum is circular, it automatically repeats the sequence once per rotation. Applications vary greatly, and a particular drum may rotate once per second, or as slowly as once per week.



Electronic drums provide the benefits of mechanical drums and more. For example, they have a **preset** feature that is impossible for mechanical drums: The preset function lets you move from the present step *directly* to any other step on command!

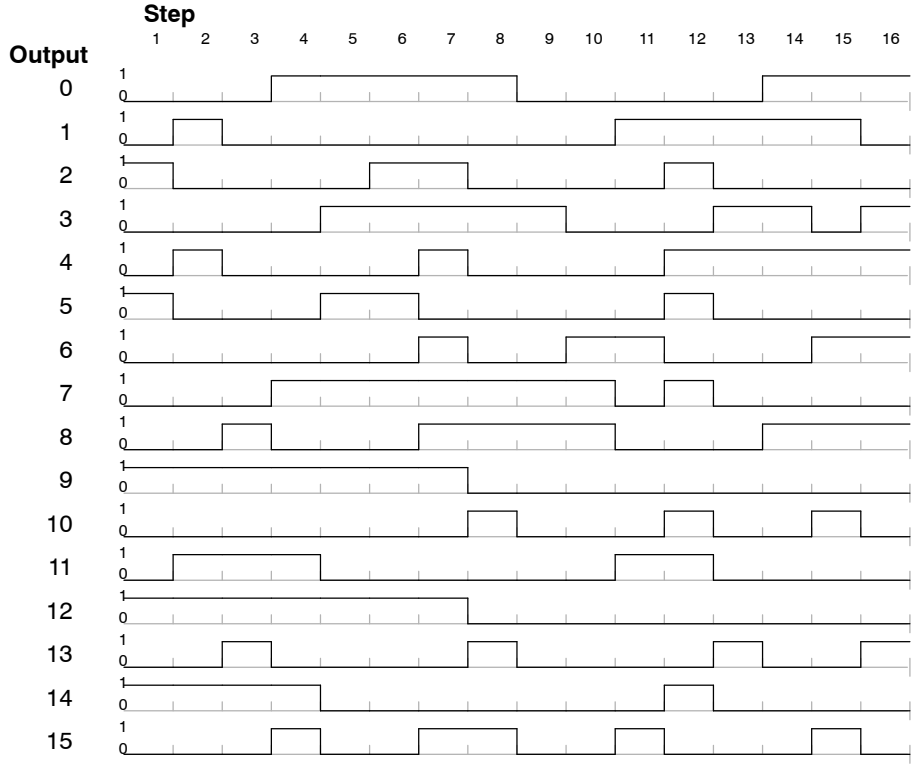
Drum Chart Representation

For editing purposes, the electronic drum is presented in chart form in ***Direct***SOFT and in this manual. Imagine slicing the surface of a hollow drum cylinder between two rows of pegs, then pressing it flat. Now you can view the drum as a chart as shown below. Each row represents a step, numbered 1 through 16. Each column represents an output, numbered 0 through 15 (to match word bit numbering). The solid circles in the chart represent pegs (On state) in the mechanical drum, and the open circles are empty peg sites (Off state).

	OUTPUTS															
STEP	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	○	●	○	●	○	○	●	○	○	○	●	○	○	●	○	○
2	○	●	○	●	●	○	●	○	○	○	○	●	○	○	●	○
3	○	●	●	●	●	○	●	●	○	○	○	○	○	○	○	○
4	●	●	○	●	●	○	●	○	●	○	○	○	○	○	○	●
5	○	○	○	●	○	○	●	○	●	○	●	○	●	○	○	●
6	○	○	○	●	○	○	●	○	●	○	●	○	●	●	○	●
7	●	○	○	●	○	○	●	●	●	●	○	●	●	●	○	●
8	●	○	●	○	○	●	○	●	●	○	○	○	●	○	○	●
9	○	○	○	○	○	○	○	●	●	○	○	○	●	○	○	○
10	○	○	○	○	○	○	○	●	●	●	○	○	○	○	○	○
11	●	○	○	○	●	○	○	○	○	●	○	○	○	○	●	○
12	○	●	○	○	●	●	○	○	●	○	●	●	○	●	●	○
13	○	○	●	○	○	○	○	○	○	○	○	●	●	○	●	○
14	○	○	○	○	○	○	○	●	○	○	○	●	●	○	●	●
15	●	○	○	○	○	●	○	●	○	●	○	●	○	○	●	●
16	○	○	●	○	○	○	○	●	○	●	○	●	●	○	○	●

Output Sequences

The mechanical drum sequencer derives its name from sequences of control changes on its electrical outputs. The following figure shows the sequence of On/Off controls generated by the drum pattern above. Compare the two, and you will find that they are equivalent! If you can see their equivalence, you are well on your way to understanding drum instruction operation.



Step Transitions

Drum Instruction Parameters

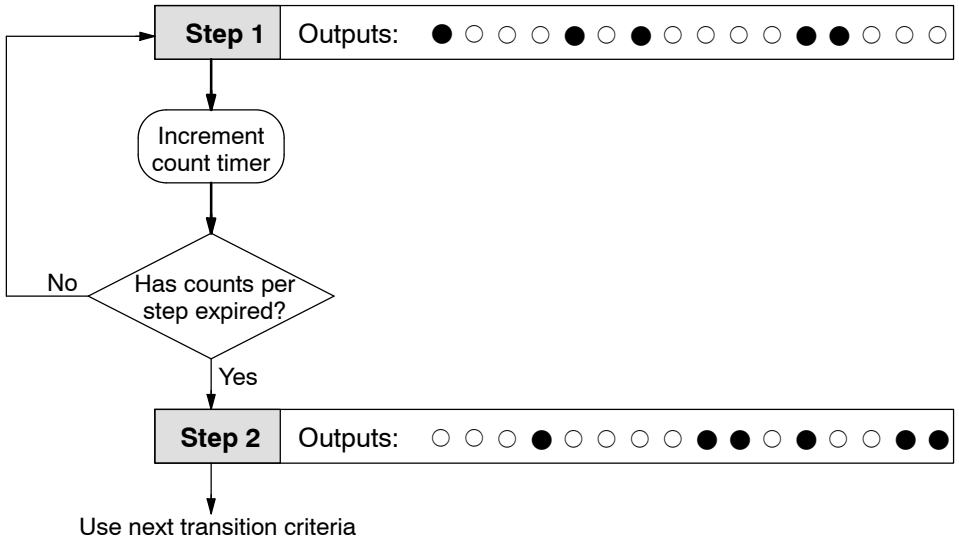
EDRUM operation in the F1-130 includes the following features:

- Up to 16 steps
- Time-based step transitions
- Event-based step transitions
- Up to 16 discrete outputs per drum (X, Y, or C type)

The EDRUM has 16 steps, and each step has 16 outputs. Refer to the figure below. Each output can be either an X, Y, or C coil, offering a lot of programming flexibility. We assign Step 1 an arbitrary unique output pattern (○= Off, ●= On) as shown. When programming the EDRUM instruction, you also determine both the output assignment and the On/Off state (pattern) at that time. All steps use the same output assignment, but each step may have its own unique output pattern.

Timer-Only Transitions

Drums move from one step to another based on time and/or an external event (input). Each step has its own transition condition which you assign during the drum instruction entry. The figure below shows how timer-only transitions work.



The drum stays in Step 1 for a specific duration (user-programmable). The timebase of the timer is programmable, from 0.01 seconds to 99.99 seconds. This establishes the resolution, or the duration of each "tick of the clock". Each step uses the same timebase, but has its own unique counts per step, which you program. When the counts for Step 1 have expired, then the drum moves to Step 2. The outputs change immediately to match the new pattern for Step 2.

The drum spends a specific amount of time in each step, given by the formula:

Time in step = 0.01 seconds X Timebase x Counts per step

For example, if you program a 5 second time base and 12 counts for Step 1, then the drum will spend 60 seconds in Step 1. The maximum time for any step is given by the formula:

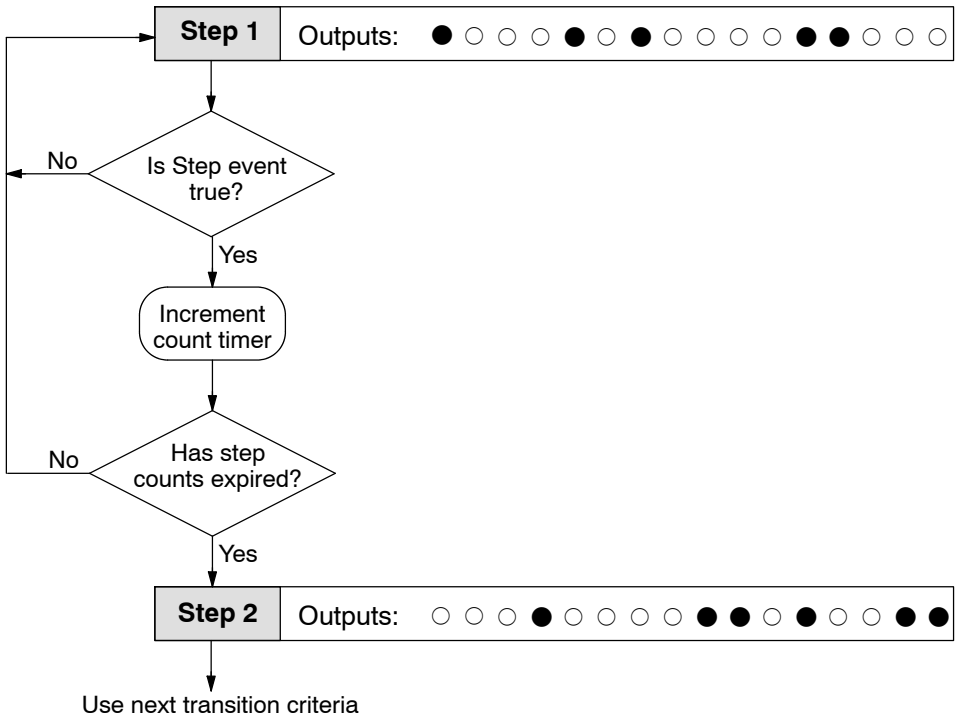
Max Time per step = 0.01 seconds X 9999 X 9999
= 999,800 seconds = 277.7 hours = 11.6 days



NOTE: When first choosing the timebase resolution, a good rule of thumb is to make it about 1/10 the duration of the shortest step in your drum. Then you will be able to optimize the duration of that step in 10% increments. Other steps with longer durations allow optimizing by even smaller increments (percentage-wise). Also, note that the drum instruction executes once per CPU scan. Therefore, it is pointless to specify a drum timebase that is much faster than the CPU scan time.

Timer and Event Transitions

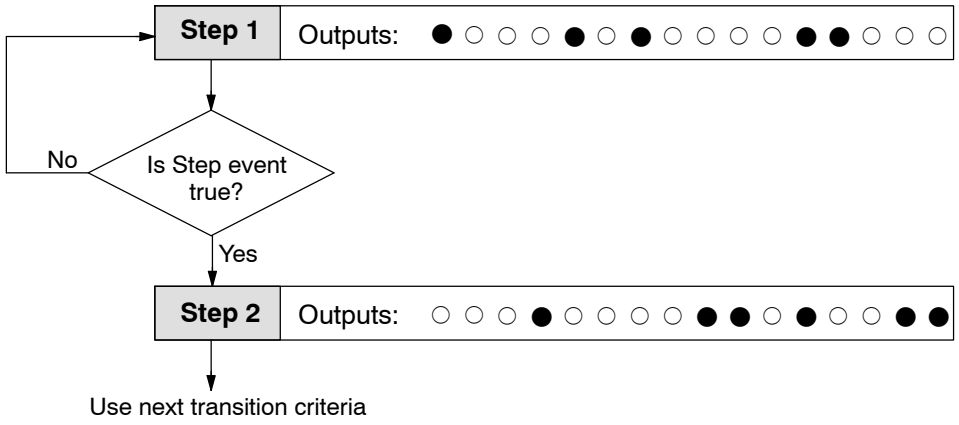
Step transitions may also occur based on time and/or external events. The figure below shows how step transitions work in these cases.



When the drum enters Step 1, it sets the output pattern as shown. Then it begins polling the external input programmed for that step. You can define event inputs as X, Y, or C discrete point types. Suppose we select X0 for the Step 1 event input. If X0 is off, then the drum remains in Step 1. When X0 is On, the event criteria is met and the timer increments. The timer increments as long as the event (X0) remains true. When the counts for Step 1 have expired, then the drum moves to Step 2. The outputs change immediately to match the new pattern for Step 2.

Event-Only Transitions

Step transitions do not require both the event and the timer criteria programmed for each step. You have the option of programming just one of the two, and even mixing transition types among all the steps of the drum. For example, you might want Step 1 to transition on an event, Step 2 to transition on time only, and Step 3 to transition on both time and an event. Furthermore, you may elect to use only part of the 16 steps, and only part of the 16 outputs.



Counter Assignments

Each drum instruction uses the resources of four counters in the CPU. When programming the drum instruction, you select the first counter number. The drum also uses the next three counters automatically. The counter bit associated with the first counter turns on when the drum has completed its cycle, going off when the drum is reset. These counter values and the counter bit precisely indicate the progress of the drum instruction, and can be monitored by your ladder program.

Suppose we program a timer drum to have 8 steps, and we select CT10 for the counter number (remember, counter numbering is in octal). Counter usage is shown to the right. The right column holds typical values, interpreted below.

Counter Assignments

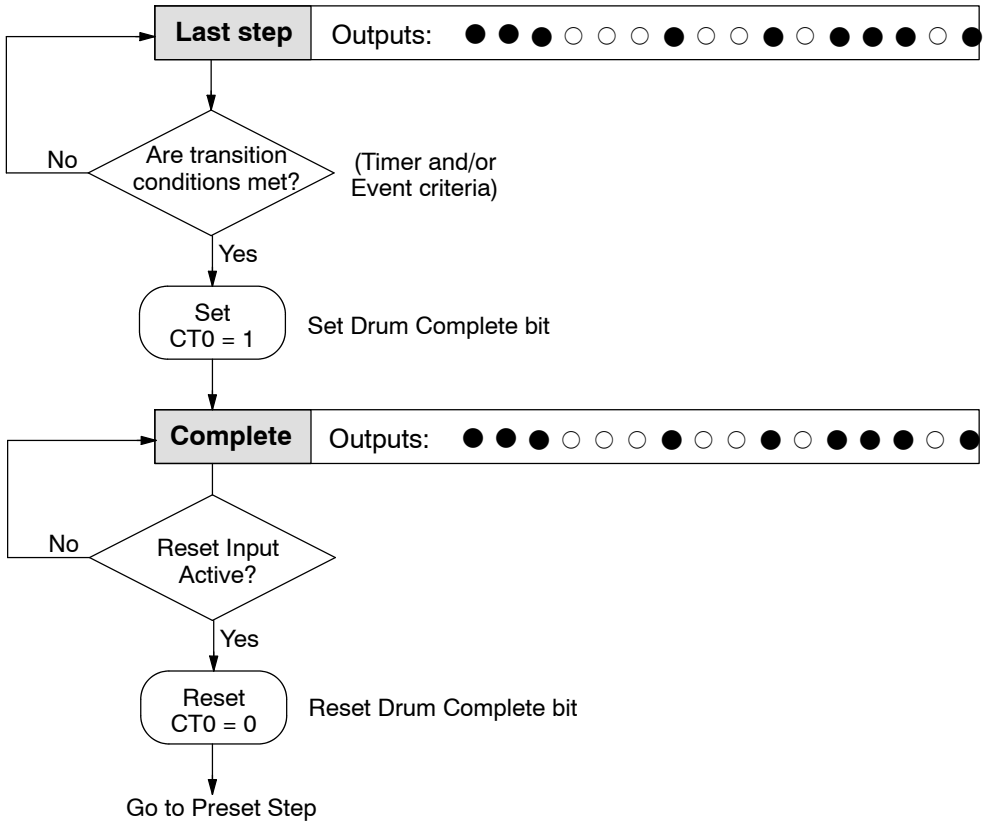
CT10	Counts in step	V1010	1528
CT11	Timer Value	V1011	0200
CT12	Preset Step	V1012	0001
CT13	Current Step	V1013	0004

CT10 shows that we are at the 1528th count in the current step, which is step 4 (shown in CT13). If we have programmed step 4 to have 3000 counts, then the step is just over half completed. CT11 is the count timer, shown in units of 0.01 seconds. So, each least-significant-digit change represents 0.01 seconds. The value of 200 means that we have been in the current count (1528) for 2 seconds (0.01 x 100). Finally, CT12 holds the preset step value which was programmed into the drum instruction. When the drum's Reset input is active, it presets to step 1 in this case. The value of CT12 changes only if the ladder program writes to it, or the drum instruction is edited and the program is restarted. Counter bit CT10 turns on when the drum cycle is complete, and turns off when the drum is reset.

Last Step Completion

The last step in a drum sequence may be any step number, since partial drums are valid. Refer to the following figure. When the transition conditions of the last step are met, the drum sets the counter bit corresponding to the counter named in the drum instruction box (such as CT0). Then it moves to a final "drum complete" state. The drum outputs remain in the pattern defined for the last step. Having finished a drum cycle, the Start and Jog inputs have no effect at this point.

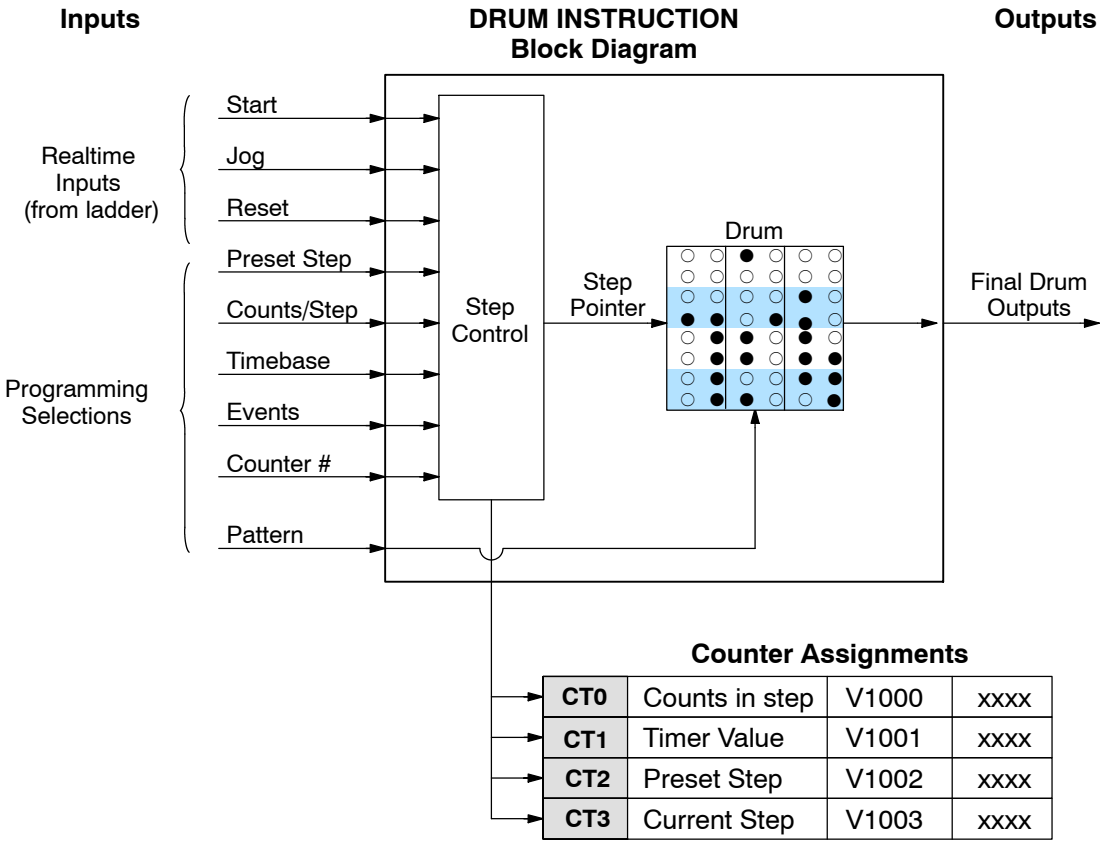
The drum leaves the "drum complete" state when the Reset input becomes active (or on a program-to-run mode transition). It resets the drum complete bit (such as CT0), and then goes directly to the appropriate step number defined as the preset step.



Overview of Drum Operation

Drum Instruction Block Diagram

The drum instruction utilizes various inputs and outputs in addition to the drum pattern itself. Refer to the figure below.



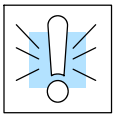
Counter Assignments

CT0	Counts in step	V1000	xxxx
CT1	Timer Value	V1001	xxxx
CT2	Preset Step	V1002	xxxx
CT3	Current Step	V1003	xxxx

The drum instruction accepts several inputs for step control, the main control of the drum. The inputs and their functions are:

- **Start** – The Start input is effective only when Reset is off. When Start is on, the drum timer runs if it is in a timed transition, and the drum looks for the input event during event transitions. When Start is off, the drum freezes in its current state (Reset must remain off), and the drum outputs maintain their current on/off pattern.
- **Jog** – The jog input is only effective when Reset is off (Start may be either on or off). The jog input increments the drum to the next step on each off-to-on transition.
- **Reset** – The Reset input has priority over the Start input. When Reset is on, the drum moves to its preset step. When Reset is off, then the Start input operates normally.
- **Preset Step** – A step number from 1 to 16 that you define (typically is step 1). The drum moves to this step whenever Reset is on, and whenever the CPU first enters run mode.

- **Counts/Step** – The number of timer counts the drum spends in each step. Each step has its own counts parameter. However, programming the counts/step is optional.
- **Timer Value** – the current value of the counts/step timer.
- **Counter #** – The counter number specifies the first of four consecutive counters which the drum uses for step control. You can monitor these to determine the drum's progress through its control cycle. The DL105 has 64 counters (CT0 – CT77 in octal).
- **Events** – Either an X, Y, C, S, T, or CT type discrete point serves as step transition inputs. Each step has its own event. However, programming the event is optional.



WARNING: The outputs of a drum are enabled any time the CPU is in Run Mode. The Start Input *does not* have to be on, and the Reset input does not disable the outputs. Upon entering Run Mode, drum outputs automatically turn on or off according to the pattern of the current step of the drum. This initial step number depends on the counter memory configuration: non-retentive versus retentive.

Powerup State of Drum Registers

The choice of the starting step on powerup and program-to-run mode transitions are important to consider for your application. Please refer to the following chart. If the counter memory is configured as non-retentive, the drum is initialized the same way on every powerup or program-to-run mode transition. However, if the counter memory is configured to be retentive, the drum will stay in its previous state.

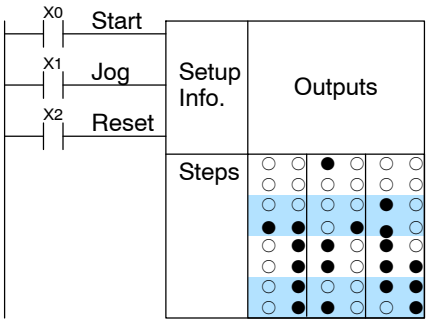
Counter Number	Function	Initialization on Powerup	
		Non-Retentive Case	Retentive Case
CT(n)	Current Step Count	Initialize = 0	Use Previous (no change)
CT(n + 1)	Counter Timer Value	Initialize = 0	Use Previous (no change)
CT(n + 2)	Preset Step	Initialize = Preset Step #	Use Previous (no change)
CT(n + 3)	Current Step #	Initialize = Preset Step #	Use Previous (no change)

Applications with relatively fast drum cycle times typically will need to be reset on powerup, using the non-retentive option. Applications with relatively long drum cycle times may need to resume at the previous point where operations stopped, using the retentive case. The default option is the retentive case. This means that if you initialize scratchpad V–memory, the memory will be retentive.

Drum Control Techniques

Drum Control Inputs

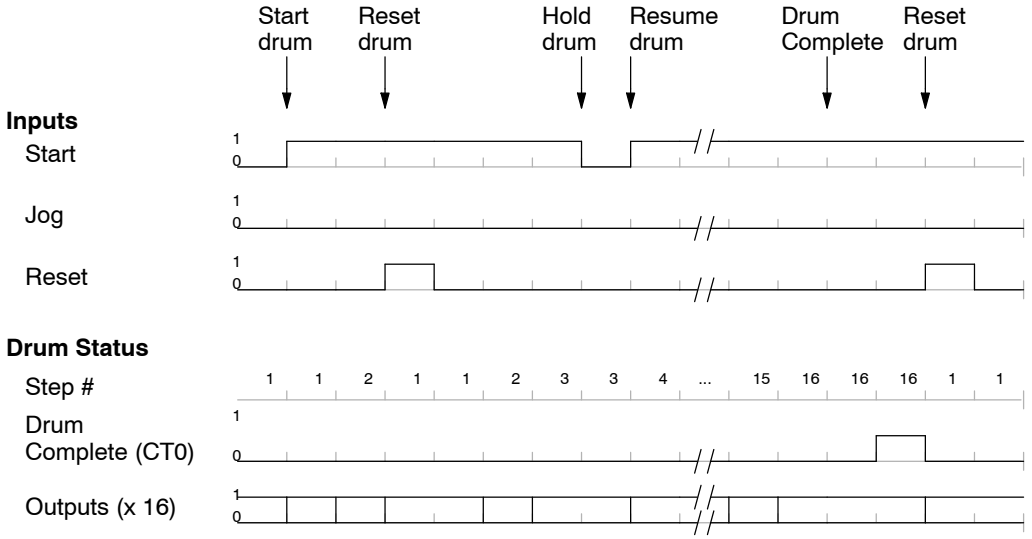
Now we are ready to put together the concepts on the previous pages and demonstrate general control of the drum instruction box. The drawing to the right shows a simplified generic drum instruction. Inputs from ladder logic control the Start, Jog, and Reset Inputs. The first counter bit of the drum (CT0, for example) indicates the drum cycle is done.



The timing diagram below shows an arbitrary timer drum input sequence and how the drum responds. As the CPU enters Run mode it initializes the step number to the preset step number (typically it is Step 1). When the Start input turns on the drum begins running, waiting for an event and/or running the timer (depends on the setup).

After the drum enters Step 2, Reset turns On while Start is still On. Since Reset has priority over Start, the drum goes to the preset step (Step 1). Note that the drum is *held* in the preset step during Reset, and that step *does not run* (respond to events or run the timer) until Reset turns off.

After the drum has entered step 3, the Start input goes off momentarily, halting the drum's timer until Start turns on again.

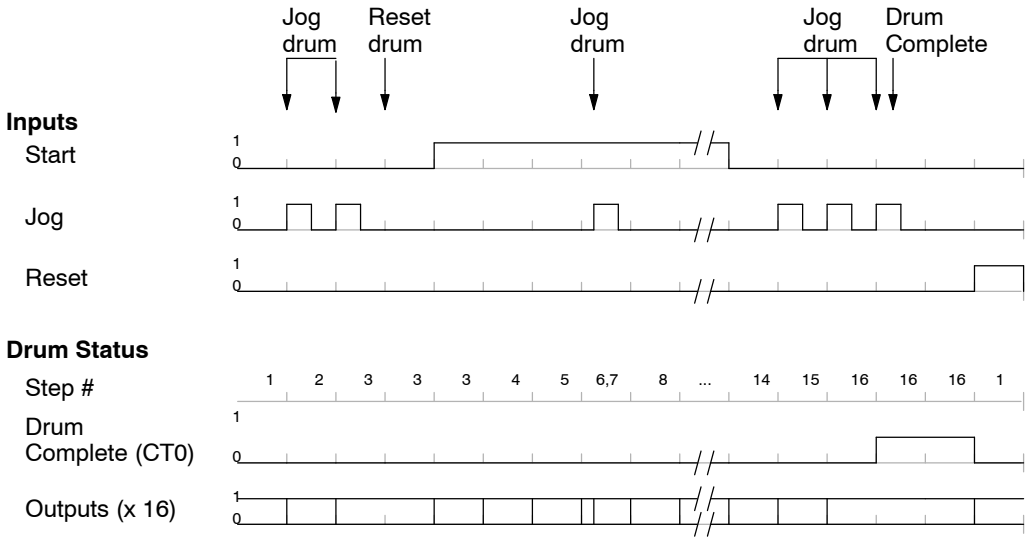


When the drum completes the last step (Step 16 in this example), the Drum Complete bit (CT0) turns on, and the step number remains at 16. When the Reset input turns on, it turns off the Drum Complete bit (CT0), and forces the drum to enter the preset step.

NOTE: The timing diagram shows all steps using equal time durations. Step times can vary greatly, depending on the counts/step programmed.

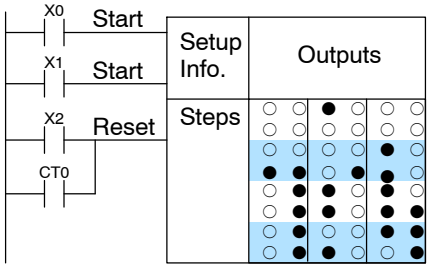
In the figure below, we focus on how the Jog input works on event drums. To the left of the diagram, note that the off-to-on transitions of the Jog input increments the step. Start may be either on or off (however, Reset must be off). Two jogs takes the drum to step three. Next, the Start input turns on, and the drum begins running normally. During step 6 another Jog input signal occurs. This increments the drum to step 7, setting the timer to 0. The drum begins running immediately in step 7, because Start is already on. The drum advances to step 8 normally.

As the drum enters step 14, the Start input turns off. Two more Jog signals moves the drum to step 16. However, note that a third Jog signal is required to move the drum through step 16 to "drum complete". Finally, a Reset input signal arrives which forces the drum into the preset step and turns off the drum complete bit.



Self-Resetting Drum

Applications often require drums that automatically start over once they complete a cycle. This is easily accomplished, using the drum complete bit. In the figure to the right, the drum instruction setup is for CT0, so we logically OR the drum complete bit (CT0) with the Reset input. When the last step is done, the drum turns on CT0 which resets itself to the preset step, also resetting CT0. Contact X2 still works as a manual reset.



Initializing Drum Outputs

The outputs of a drum are enabled any time the CPU is in run mode. On program-to-run mode transitions, the drum goes to the preset step, and the outputs energize according to the pattern of that step. If your application requires all outputs to be off at powerup, make the preset step in the drum a "reset step", with all outputs off.

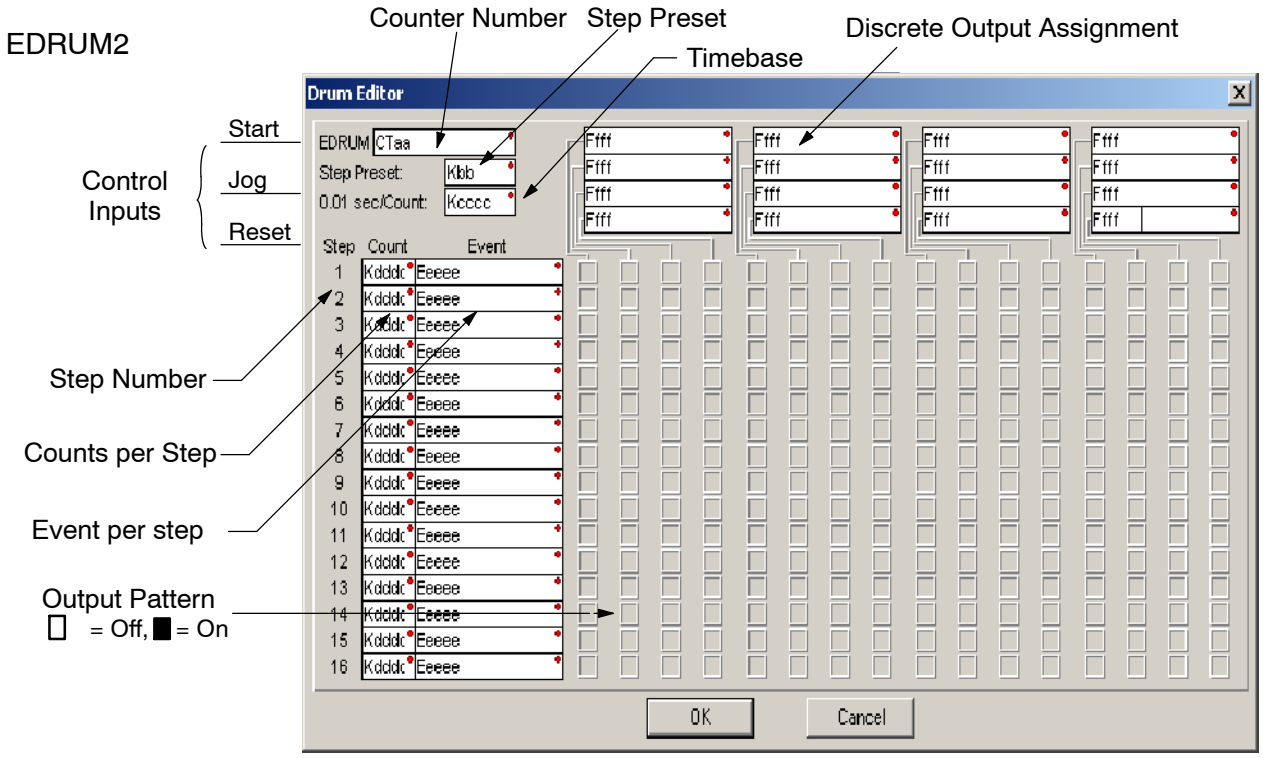
Using Complex Event Step Transitions

Each event-based transition accepts only one contact reference for the event. However, this does not limit events to just one contact. Just use a control relay contact such as C0 for the step transition event. Elsewhere in ladder logic, you may use C0 as an output coil, making it dependent on many other "events" (contacts).

Drum Instruction

Event Drum (EDRUM)

The Event Drum (EDRUM) features time-based and event-based step transitions. It operates according to the general principles of drum operation covered in the beginning of this chapter. Below is the instruction as displayed by ***Direct***SOFT.



The Event Drum features 16 steps and 16 discrete outputs. Step transitions occur on timed and/or event basis. The jog input also advances the step on each off-to-on transition. Time is specified in counts per step, and events are specified as discrete contacts. Unused steps must be programmed with "counts per step" = 0, and event = "K0000". The discrete output points may be individually assigned.

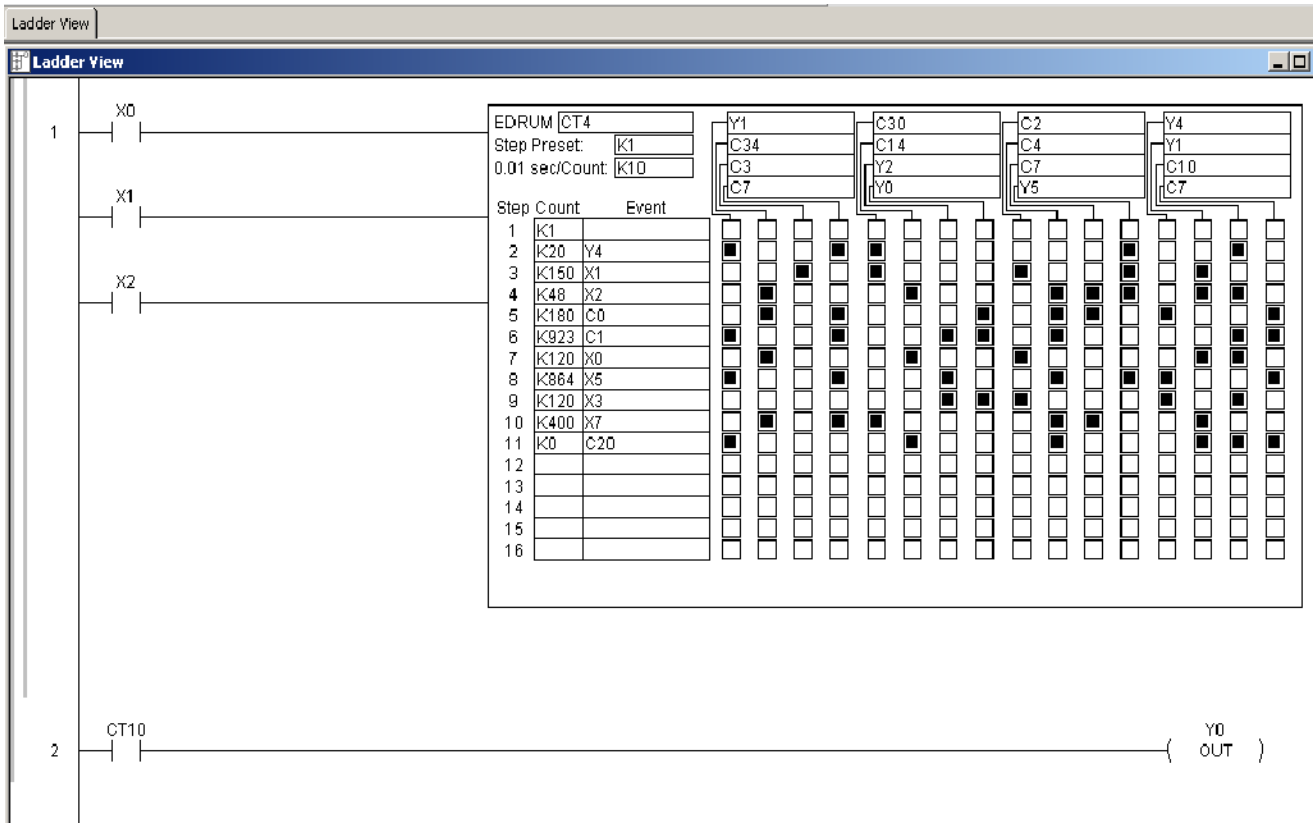
Whenever the Start input is energized, the drum's timer is enabled. As long as the event is true for the current step, the timer runs during that step. When the step count equals the counts per step, the drum transitions to the next step. This process stops when the last step is complete, or when the Reset input is energized. The drum enters the preset step chosen upon a CPU program-to-run mode transition, and whenever the Reset input is energized.

Drum Parameters	Field	Data Types	Ranges
Counter Number	aa	-	0 - 74
Preset Step	bb	K	1 - 16
Timer base	cccc	K	0.01 - 99.99 seconds
Counts per step	dddd	K	0 - 9999
Event	eeee	X, Y, C, S, T, CT	see page 4-29
Discrete Outputs	ffff	X, Y, C	see page 4-29

Drum instructions use four counters in the CPU. The ladder program can read the counter values for the drum's status. The ladder program may write a new preset step number to CT(n+2) at any time. However, the other counters are for monitoring purposes only.

Counter Number	Ranges of (n)	Function	Counter Bit Function
CT(n)	0 – 74	Counts in step	CTn = Drum Complete
C(n+1)	1 – 75	Timer value	CT(n+1) = (not used)
CT(n+2)	2 –76	Preset Step	CT(n+2) = (not used)
CT(n+3)	3 –77	Current Step	CT(n+1) = (not used)

The following ladder program shows the EDRUM instruction in a typical ladder program, as shown by ***Direct***SOFT. Steps 1 through 11 are used, and all sixteen output points are used. The preset step is step 1. The timebase runs at (K100 x 0.01) = 0.1 second per count. Therefore, the duration of step 1 is (5 x 0.1) = 0.5 seconds. Note that step 1 is time-based only (event = "K0000"). And, the output pattern for step 1 programs all outputs off, which is a typically desirable powerup condition. In the last rung, the Drum Complete bit (CT4) turns on output Y0 upon completion of the last step (step 10). A drum reset also resets CT4.

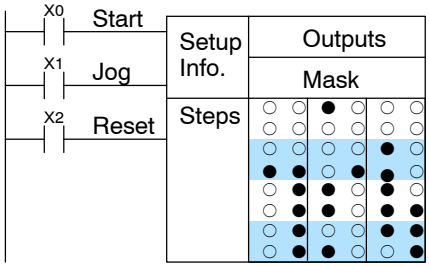


Handheld Programer Drum Mnemonics

The EDRUM instruction may be programmed using either ***Direct***SOFT or a handheld programmer. This section covers entry via the handheld programmer (Refer to the ***Direct***SOFT manual for drum instruction entry using that tool).

First, enter Store instructions for the ladder rungs controlling the drum's ladder inputs. In the example to the right, the timer drum's Start, Jog, and Reset inputs are controlled by X0, X1 and X2 respectively. The required keystrokes are listed beside the mnemonic.

These keystrokes *precede* the EDRUM instruction mnemonic. Note that the ladder rungs for Start, Jog, and Reset inputs are *not* limited to being single-contact rungs.



Handheld Programmer Keystrokes

Store X0 [$ STR] [→] [A 0] [ENT]

(Repeat for Store X1 and Store X2)

After the Store instructions, enter the EDRUM (using Counter CT0) as shown:

Handheld Programmer Keystrokes

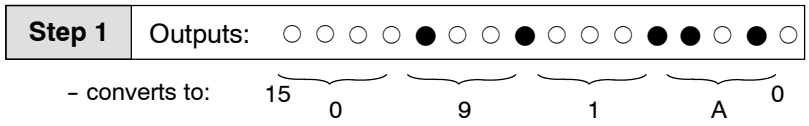
EDRUM CNT0 [SHFT] [E 4] [D 3] [R ORN] [U ISG] [M ORST] [→] [A 0] [ENT]

After entering the EDRUM mnemonic as above, the handheld programmer creates an input form for all the drum parameters. The input form consists of approximately fifty or more default mnemonic entries containing DEF (define) statements. The default mnemonics are already "input" for you, so they appear automatically. Use the NXT and PREV keys to move forward and backward through the form. Only the editing of default values is required, thus eliminating many keystrokes. The entries required for the basic timer drum are in the chart below.

Drum Parameters	Multi-ple En-tries	Mnemonic / Entry	Default Mnemonic	Valid Data Types	Ranges
Start Input	–	STR (plus input rung)	–	–	–
Jog Input	–	STR (plus input rung)	–	–	–
Reset Input	–	STR (plus input rung)		–	–
Drum Mnemonic	–	DRUM CNT aa	–	K	0 – 74
Preset Step	1	bb	DEF K0000	K	1 – 16
Timer base	1	cccc	DEF K0000	K	2 – 9999
Output points	16	ffff	DEF 0000	X, Y, C *	see page 4-29
Counts per step	16	dddd	DEF K0000	K	0 – 9999
Events	16	dddd	DEF K0000	X, Y, C, S, T, CT	see page 4-29
Output pattern	16	gggg	DEF K0000	K	0 – FFFF

NOTE: Default entries for output points and events are "DEF 0000", which means they are unassigned. If you need to go back and change an assigned output as unused again, enter "K0000". The entry will again show as "DEF 0000".

Using the DRUM entry chart (two pages before), we show the method of entry for the basic time/event drum instruction. First, we convert the output pattern for each step to the equivalent hex number, as shown in the following example.



The following diagram shows the method for entering the previous EDRUM example on the HHP. The default entries of the form are in parenthesis. After the drum instruction entry (on the fourth row), the remaining keystrokes over-write the numeric portion of each default DEF statement. **NOTE**: Drum editing requires Handheld Programmer firmware version 1.7 or later.

Handheld Programmer Keystrokes

Start		$ STR → A 0 ENT
Jog		$ STR → B 1 ENT
Reset		$ STR → C 2 ENT
Drum Inst.		SHFT E 4 D 3 R ORN U ISG M ORST → A 0 ENT
Preset Step	(DEF K0001)	NEXT
Time Base	(DEF K0000)	G 6 E 4 NEXT
Outputs	1 (DEF 0000)	SHFT C 2 H 7 NEXT
	(DEF 0000)	SHFT C 2 B 1 A 0 NEXT
	(DEF 0000)	SHFT Y MLS B 1 NEXT
	(DEF 0000)	SHFT Y MLS E 4 NEXT
	(DEF 0000)	SHFT Y MLS F 5 NEXT
	(DEF 0000)	SHFT Y MLS G 6 NEXT
	(DEF 0000)	SHFT C 2 E 4 NEXT
	(DEF 0000)	SHFT C 2 C 2 NEXT
	(DEF 0000)	SHFT Y MLS A 0 NEXT
	(DEF 0000)	SHFT Y MLS C 2 NEXT
	(DEF 0000)	SHFT C 2 B 1 E 4 NEXT
	(DEF 0000)	SHFT C 2 D 3 A 0 NEXT
	(DEF 0000)	SHFT Y MLS G 6 NEXT
	(DEF 0000)	SHFT Y MLS H 7 NEXT
	(DEF 0000)	SHFT C 2 D 3 E 4 NEXT
	16 (DEF 0000)	SHFT Y MLS B 1 NEXT

NOTE: You may use the NXT and PREV keys to skip past entries for unused outputs or steps.

Handheld Programmer Keystrokes cont'd

Counts/ Step	1 (DEF K0000)	F 5 NEXT
	(DEF K0000)	C 2 A 0 NEXT
	(DEF K0000)	B 1 F 5 A 0 NEXT
	(DEF K0000)	E 4 F 5 NEXT
	(DEF K0000)	B 1 I 8 A 0 NEXT
	(DEF K0000)	J 9 C 2 D 3 NEXT
	(DEF K0000)	B 1 C 2 A 0 NEXT
	(DEF K0000)	I 8 G 6 E 4 NEXT
	(DEF K0000)	B 1 C 2 A 0 A 0 NEXT
	(DEF K0000)	E 4 A 0 A 0 NEXT
	(DEF K0000)	NEXT
	(DEF K0000)	NEXT
	(DEF K0000)	NEXT ← skip over unused steps
	(DEF K0000)	NEXT
	(DEF K0000)	NEXT
	16 (DEF K0000)	NEXT

(Continued on next page)

Handheld Programmer Keystrokes cont'd

Events

1	(DEF 0000)	NEXT	← skip over unused event			
	(DEF 0000)	SHFT	Y MLS	E 4	NEXT	
	(DEF 0000)	SHFT	X SET	B 1	NEXT	
	(DEF 0000)	SHFT	X SET	C 2	NEXT	
	(DEF 0000)	SHFT	C 2	A 0	NEXT	
	(DEF 0000)	SHFT	C 2	B 1	NEXT	
	(DEF 0000)	SHFT	X SET	A 0	NEXT	
	(DEF 0000)	SHFT	X SET	F 5	NEXT	
	(DEF 0000)	SHFT	X SET	D 3	NEXT	
	(DEF 0000)	SHFT	Y MLS	H 7	NEXT	
	(DEF 0000)	SHFT	C 2	C 2	A 0	NEXT
	(DEF 0000)	NEXT				
	(DEF 0000)	NEXT				
	(DEF 0000)	NEXT				
	(DEF 0000)	NEXT				
16	(DEF 0000)	NEXT				

Handheld Programmer Keystrokes cont'd

Output Pattern

1	(DEF K0000)	NEXT	← step 1 pattern = 0000			
	(DEF K0000)	J 9	I 8	B 1	C 2	NEXT
	(DEF K0000)	C 2	I 8	J 9	E 4	NEXT
	(DEF K0000)	E 4	E 4	H 7	G 6	NEXT
	(DEF K0000)	F 5	B 1	G 6	J 9	NEXT
	(DEF K0000)	J 9	D 3	E 4	D 3	NEXT
	(DEF K0000)	E 4	E 4	I 8	G 6	NEXT
	(DEF K0000)	J 9	E 4	F 5	J 9	NEXT
	(DEF K0000)	D 3	I 8	SHFT	A 0	NEXT
	(DEF K0000)	F 5	I 8	G 6	E 4	NEXT
	(DEF K0000)	I 8	E 4	E 4	H 7	NEXT
	(DEF K0000)	NEXT				
	(DEF K0000)	NEXT				
	(DEF K0000)	NEXT	← unused steps			
	(DEF K0000)	NEXT				
16	(DEF K0000)	NEXT				

Last rung

$ STR	GY CNT	A 0	NEXT
SHFT	Y MLS	A 0	NEXT

NOTE: You may use the NXT and PREV keys to skip past entries for unused outputs or steps.

7

RLL*PLUS* Stage Programming

In This Chapter. . . .

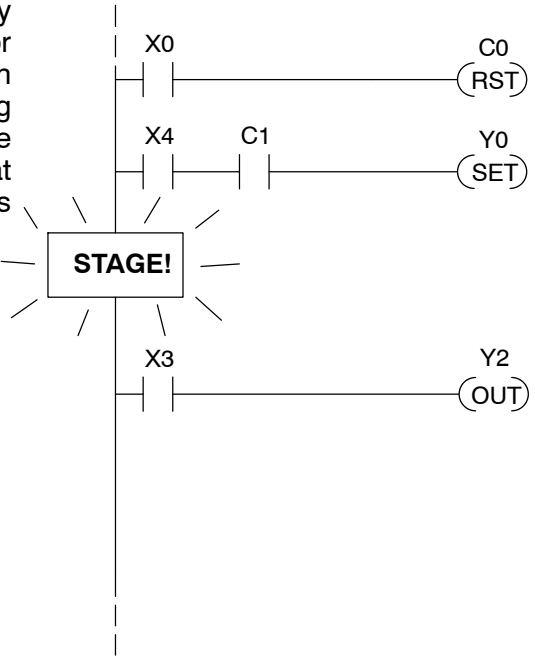
— Introduction to Stage Programming
— Learning to Draw State Transition Diagrams
— Using the Stage Jump Instruction for State Transitions
— Stage Program Example: Toggle On/Off Lamp Controller
— Four Steps to Writing a Stage Program
— Stage Program Example: A Garage Door Opener
— Stage Program Design Considerations
— RLL*PLUS* Stage Instructions
— Questions and Answers about Stage Programs

Introduction to Stage Programming

Stage Programming provides a way to organize and program complex applications with relative ease, when compared to purely relay ladder logic (RLL) solutions. Stage programming does not replace or negate the use of traditional boolean ladder programming. This is why Stage Programming is also called RLL*PLUS*. You won't have to discard any training or experience you already have. Stage programming simply allows you to divide and organize a RLL program into groups of ladder instructions called stages. This allows quicker and more intuitive ladder program development than traditional RLL alone provides.

Overcoming "Stage Fright"

Many PLC programmers in the industry have become comfortable using RLL for every PLC program they write... but often remain skeptical or even fearful of learning new techniques such as stage programming. While RLL is great at solving boolean logic relationships, it has disadvantages as well:



- Large programs can become almost unmanageable, because of a lack of structure.
- In RLL, latches must be tediously created from self-latching relays.
- When a process gets stuck, it is difficult to find the rung where the error occurred.
- Programs become difficult to modify later, because they do not intuitively resemble the application problem they are solving.

It's easy to see that these inefficiencies consume a lot of additional time, and time is money. *Stage programming overcomes these obstacles!* We believe a few moments of studying the stage concept is one of the greatest investments in programming speed and efficiency a PLC programmer can make!

So, we encourage you to study stage programming and add it to your "toolbox" of programming techniques. This chapter is designed as a self-paced tutorial on stage programming. For best results:

- Start at the beginning and do not skip over any sections.
- Study each stage programming concept by working through each example. The examples build progressively on each other.
- Read the stage Questions and Answers at the end of the chapter for a quick review.

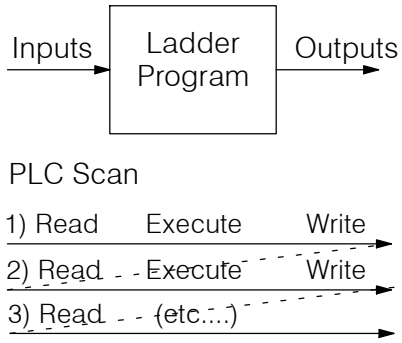
Learning to Draw State Transition Diagrams

Introduction to Process States

Those familiar with ladder program execution know that the CPU must scan the ladder program repeatedly, over and over. Its three basic steps are:

1. Read the inputs
2. Execute the ladder program
3. Write the outputs

The benefit is that a change at the inputs can affect the outputs in just a few milliseconds.



Most manufacturing processes consist of a series of activities or conditions , each lasting for several seconds. minutes, or even hours. We might call these “process states”, which are either active or inactive at any particular time. A challenge for RLL programs is that a particular input event may last for just a brief instant. We typically create latching relays in RLL to preserve the input event in order to maintain a process state for the required duration.

We can organize and divide ladder logic into sections called “stages”, representing process states. But before we describe stages in detail, we will reveal **the secret to understanding stage programming:** state transition diagrams.

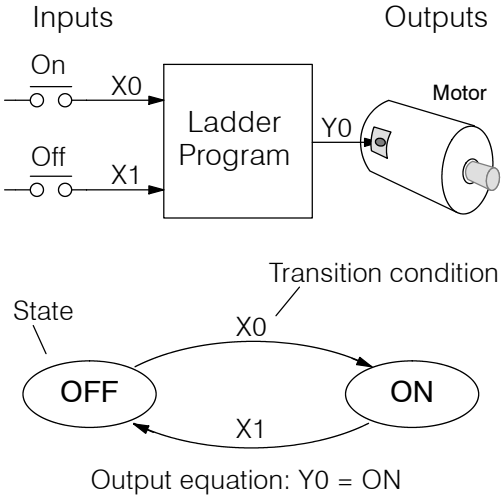
The Need for State Diagrams

Sometimes we need to forget about the scan nature of PLCs, and focus our thinking toward the states of the process we need to identify. Clear thinking and concise analysis of an application gives us the best chance at writing efficient, bug-free programs. *State diagrams are just a tool to help us draw a picture of our process!* You’ll discover that if we can get the picture right, **our program will also be right!**

A 2-State Process

Consider the simple process shown to the right, which controls an industrial motor. We will use a green momentary SPST pushbutton to turn the motor on, and a red one to turn it off. The machine operator will press the appropriate pushbutton for just a second or so. The two states of our process are ON and OFF.

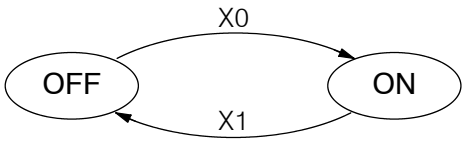
The next step is to draw a *state transition diagram*, as shown to the right. It shows the two states OFF and ON, with two transition lines in-between. When the event X0 is true, we transition from OFF to ON. When X1 is true, we transition from ON to OFF.



If you’re following along, you are very close to grasping the concept and the problem-solving power of state transition diagrams. The output of our controller is Y0, which is true any time we are in the ON state. In a boolean sense, Y0=ON state.

Next, we will implement the state diagram first as RLL, then as a stage program. This will help you see the relationship between the two methods in problem solving.

The state transition diagram to the right is a picture of the solution we need to create. The beauty of it is this: it expresses the problem independently of the programming language we may use to realize it. In other words, *by drawing the diagram we have already solved the control problem!*



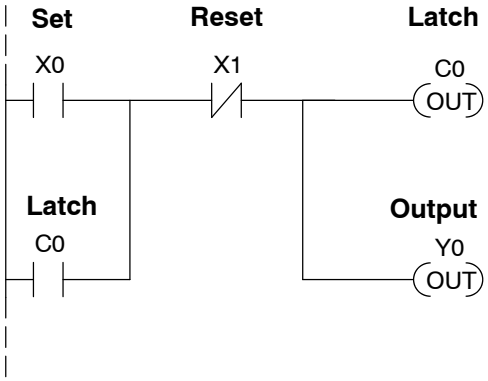
Output equation: Y0 = ON

First, we'll translate the state diagram to traditional RLL. Then we'll show how easy it is to translate the diagram into a stage programming solution.

RLL Equivalent

The RLL solution is shown to the right. It consists of a self-latching control relay, C0. When the On pushbutton (X0) is pressed, output coil C0 turns on and the C0 contact on the second row latches itself on. So, X0 **sets the latch** C0 on, and it remains on after the X0 contact opens. The motor output Y0 also has power flow, so the motor is now on.

When the Off pushbutton (X1) is pressed, it opens the normally-closed X1 contact, which **resets the latch**. Motor output Y0 turns off when the latch coil C0 goes off.

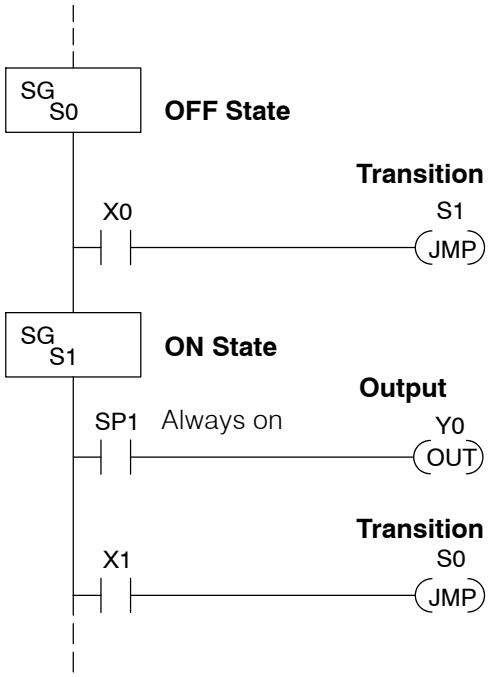


Stage Equivalent

The stage program solution is shown to the right. The two inline stage boxes S0 and S1 correspond to the two states OFF and ON. The ladder rung(s) below each stage box belong to each respective stage. This means that the PLC only has to scan those rungs when the corresponding stage is active!

For now, let's assume we begin in the OFF State, so stage S0 is active. When the On pushbutton (X0) is pressed, a stage transition occurs. The JMP S1 instruction executes, which simply turns off the stage bit S0 and turns on stage bit S1. So on the next PLC scan, the CPU will not execute stage S0, but will execute stage S1!

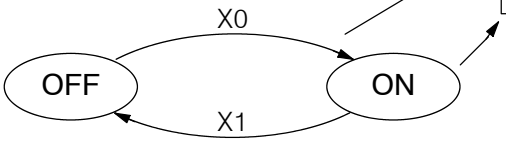
In the On State (stage S1), we want the motor to always be on. The special relay contact SP1 is defined as always on, so Y0 turns the motor on.

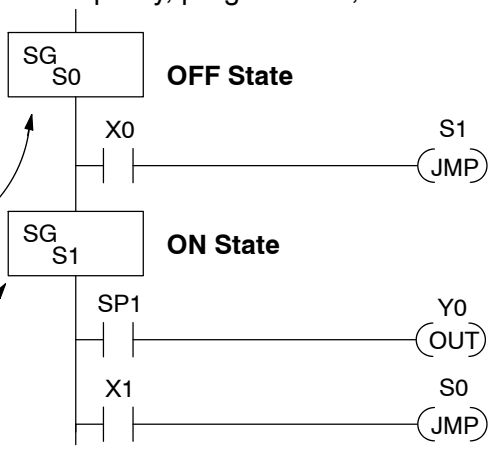


When the Off pushbutton (X1) is pressed, a transition back to the Off State occurs. The JMP S0 instruction executes, which simply turns off the stage bit S1 and turns on stage bit S0. On the next PLC scan, the CPU will not execute stage S1, so the motor output Y0 will turn off. The Off state (stage 0) will be ready for the next cycle.

Let's Compare

Right now, you may be thinking "I don't see the big advantage to Stage Programming... in fact, the stage program is longer than the plain RLL program". Well, now is the time to exercise a bit of faith. As control problems grow in complexity, stage programming quickly out-performs RLL in simplicity, program size, etc. For example, consider the diagram below. Notice how easy it is to correlate the OFF and ON states of the state transition diagram below to the stage program at the right. Now, we challenge anyone to easily identify the same states in the RLL program on the previous page!

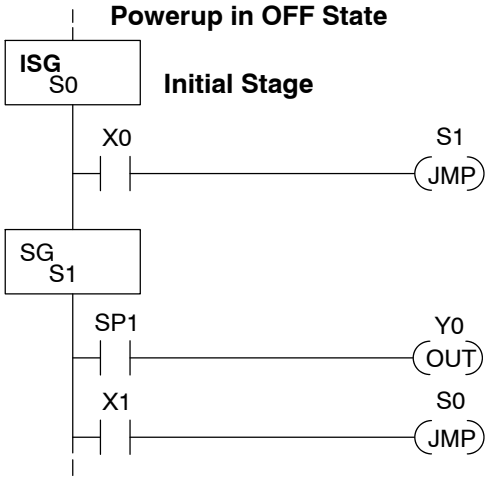




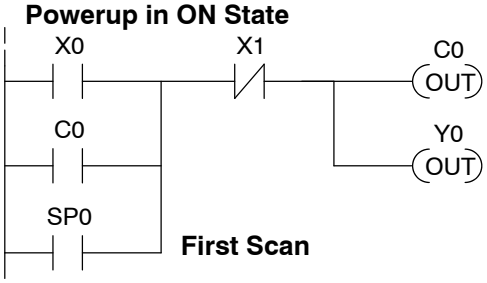
Initial Stages

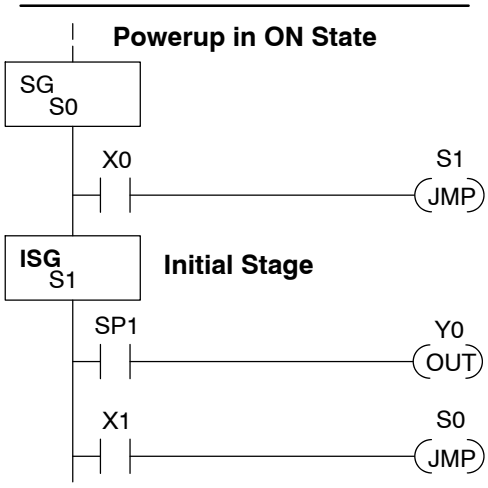
At powerup and Program-to-Run Mode transitions, the PLC always begins with all normal stages (SG) off. So, the stage programs shown so far have actually had no way to get started (because rungs are not scanned unless their stage is active).

Assume that we want to always begin in the Off state (motor off), which is how the RLL program works. The Initial Stage (ISG) is defined to be active at powerup. In the modified program to the right, we have changed stage S0 to the ISG type. This ensures the PLC will scan contact X0 after powerup, because Stage S0 is active. **After powerup, an Initial Stage (ISG) works just like any other stage!**



We can change both programs so that the motor is ON at powerup. In the RLL below, we must add a first scan relay SP0, latching C0 on. In the stage example to the right, we simply make Stage S1 an initial stage (ISG) instead of S0.

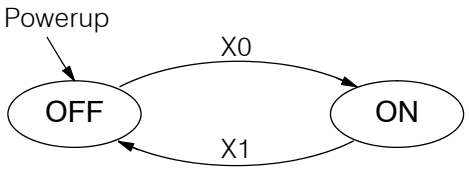






NOTE: If the ISG is within the retentive range for stages, the ISG will remain in the state it was in before power down and will NOT turn itself on during the first scan.

We can mark our desired powerup state as shown to the right, which helps us remember to use the appropriate Initial Stages when creating a stage program. It is permissible to have as many initial stages as the process requires.

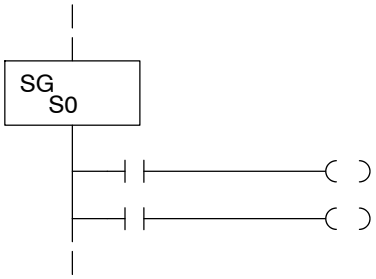


What Stage Bits Do You may recall that a stage is just a section of ladder program which is either active or inactive at a given moment. All stage bits (S0 to S77) reside in the PLC's image register as individual status bits. Each stage bit is either a boolean 0 or 1 at any time.

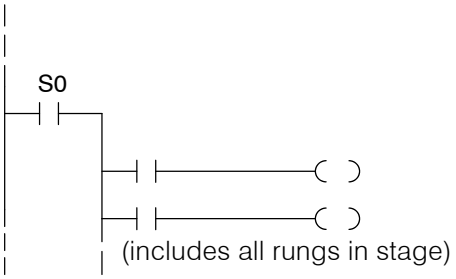
Program execution always reads ladder rungs from top to bottom, and from left to right. The drawing below shows the effect of stage bit status. The ladder rungs below the stage instruction continuing until the next stage instruction or the end of program belong to stage 0. Its equivalent operation is shown on the right. When S0 is true, the two rungs have power flow.

- If Stage bit S0 = 0, its ladder rungs *are not* scanned (executed).
- If stage bit S0 = 1, its ladder rungs *are* scanned (executed).

Actual Program Appearance

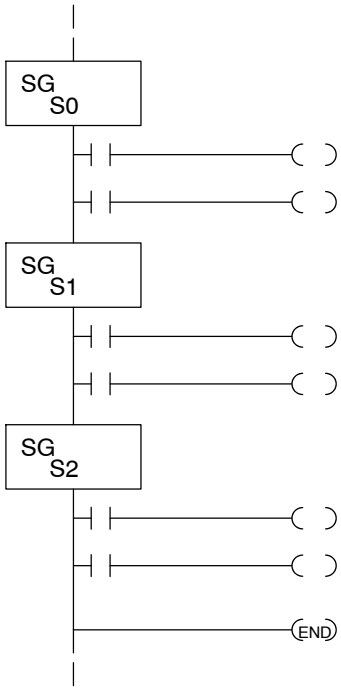


Functionally Equivalent Ladder



Stage Instruction Characteristics The inline stage boxes on the left power rail divide the ladder program rungs into stages. Some stage rules are:

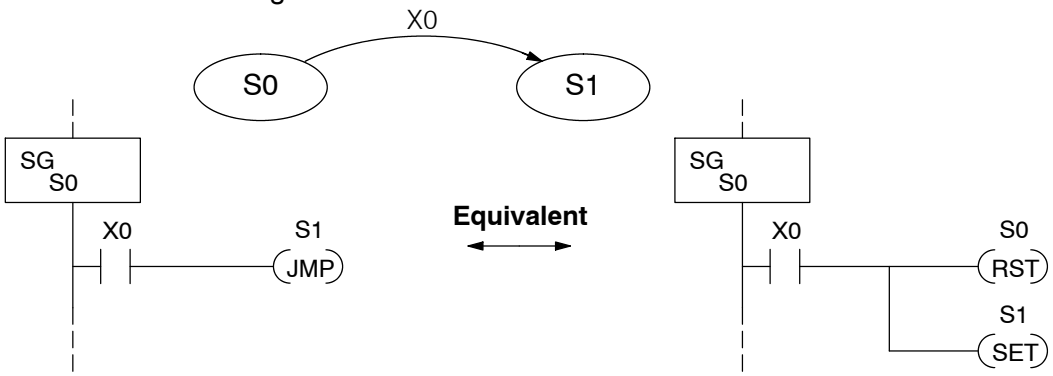
- **Execution** – Only logic in active stages are executed on any scan.
- **Transitions** – Stage transition instructions take effect on the next occurrence of the stages involved.
- **Octal numbering** – Stages are numbered in octal, like I/O points, etc. So "S8" is not valid.
- **Total Stages** – The DL105 offers up to 256 stages (S0 to S377 in octal).
- **No duplicates** – Each stage number is unique and can be used just once.
- **Any order** – You can skip numbers and sequence the stage numbers in any order.
- **Last Stage** – the last stage in the ladder program includes all rungs from its stage box until the end coil.



Using the Stage Jump Instruction for State Transitions

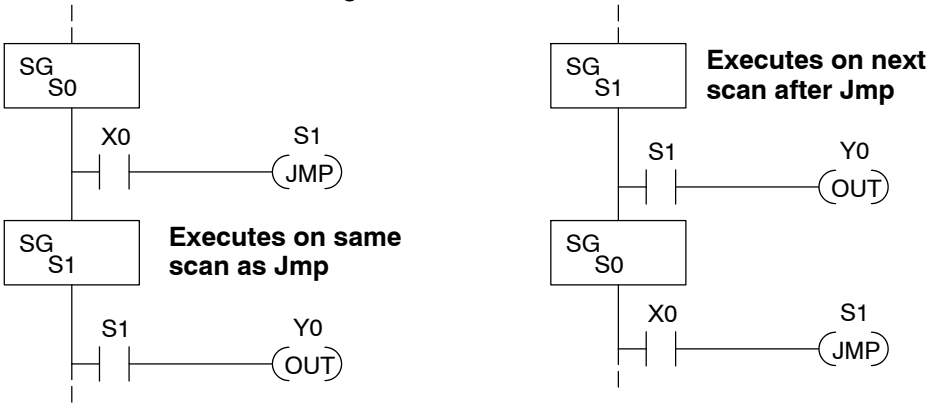
Stage Jump, Set, and Reset Instructions

The stage JMP instruction we have used deactivates the stage in which the instruction occurs, while activating the stage in the JMP instruction. Refer to the state transition shown below. When contact X0 energizes, the state transition from S0 to S1 occurs. The two stage examples shown below are equivalent. So, the stage Jump instruction is equal to a Stage Reset of the current stage, plus a Stage Set instruction for the stage to which we want to transition.



Please Read Carefully – The jump instruction is easily misunderstood. The "jump" does not occur immediately like a GOTO or GOSUB program control instruction when executed. Here's how it works:

- The jump instruction resets the stage bit of the stage in which it occurs. All rungs in the stage still finish executing during the current scan, *even if there are other rungs in the stage below the jump instruction!*
- The reset will be in effect on the following scan, so the stage that executed the jump instruction previously will be inactive and bypassed.
- The stage bit of the stage named in the Jump instruction will be set immediately, so the stage will be executed on its next occurrence. In the left program shown below, stage S1 executes during the *same scan* as the JMP S1 occurs in S0. In the example on the right, Stage S1 executes on the *next scan* after the JMP S1 executes, because stage S1 is located *above* stage S0.

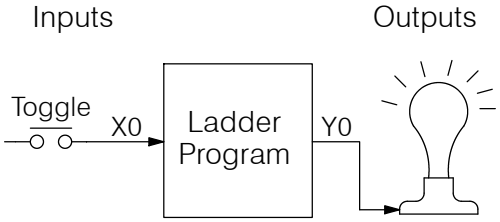


NOTE: Assume we start with Stage 0 active and Stage 1 inactive for both examples.

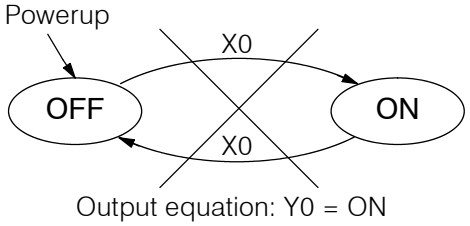
Stage Program Example: Toggle On/Off Lamp Controller

A 4-State Process

In the process shown to the right, we use an ordinary momentary pushbutton to control a light bulb. The ladder program will latch the switch input, so that we will push and release to turn on the light, push and release again to turn it off (sometimes called toggle function). Sure, we could just buy a mechanical switch with the alternate on/off action built in... However, this example is educational and also fun!

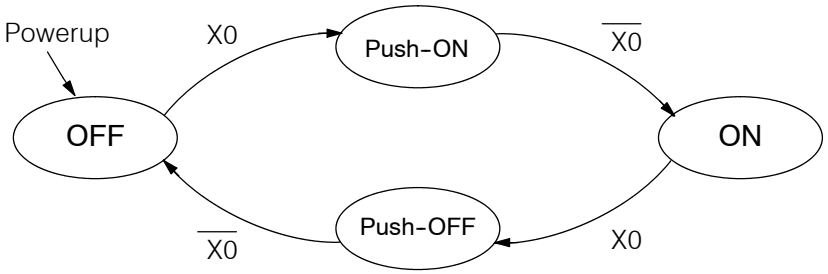


Next we draw the state transition diagram. A typical first approach is to use X0 for both transitions (like the example shown to the right). However, *this is incorrect* (please keep reading).



Note that this example differs from the motor example, because now we have just one pushbutton. When we press the pushbutton, both transition conditions are met. We would just transition around the state diagram at top speed. If implemented in stage, this solution would flash the light on or off each scan (obviously undesirable)!

The solution is to make the the push and the release of the pushbutton separate events. Refer to the new state transition diagram below. At powerup we enter the OFF state. When switch X0 is pressed, we enter the Press-ON state. When it is released, we enter the ON state. Note that X0 with the bar above it denotes X0 NOT.

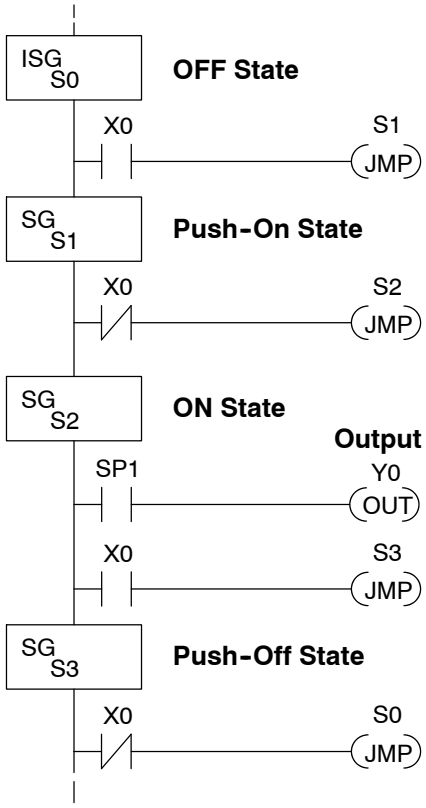


Output equation: Y0 = ON

When in the ON state, another push and release cycle similarly takes us back to the OFF state. Now we have two unique states (OFF and ON) used when the pushbutton is released, which is what was required to solve the control problem.

The equivalent stage program is shown to the right. The desired powerup state is OFF, so we make S0 an initial stage (ISG). In the ON state, we add special relay contact SP1, which is always on.

Note that even as our programs grow more complex, it is still easy to correlate the state transition diagram with the stage program!



Four Steps to Writing a Stage Program

By now, you've probably noticed that we follow the same steps to solve each example problem. The steps will probably come to you automatically if you work through all the examples in this chapter. It's helpful to have a checklist to guide us through the problem solving. The following steps summarize the stage program design procedure:

1. Write a Word Description of the application.

Describe all functions of the process in your own words. Start by listing what happens first, then next, etc. If you find there are too many things happening at once, try dividing the problem into more than one process. Remember, you can still have the processes communicate with each other to coordinate their overall activity.

2. Draw the Block Diagram.

Inputs represent all the information the process needs for decisions, and outputs connect to all devices controlled by the process.

- Make lists of inputs and outputs for the process.
- Assign I/O point numbers (X and Y) to physical inputs and outputs.

3. Draw the State Transition Diagram.

The state transition diagram describes the central function of the block diagram, reading inputs and generating outputs.

- Identify and name the states of the process.
- Identify the event(s) required for each transition between states.
- Ensure the process has a way to re-start itself, or is cyclical.
- Choose the powerup state for your process.
- Write the output equations.

4. Write the Stage Program.

Translate the state transition diagram into a stage program.

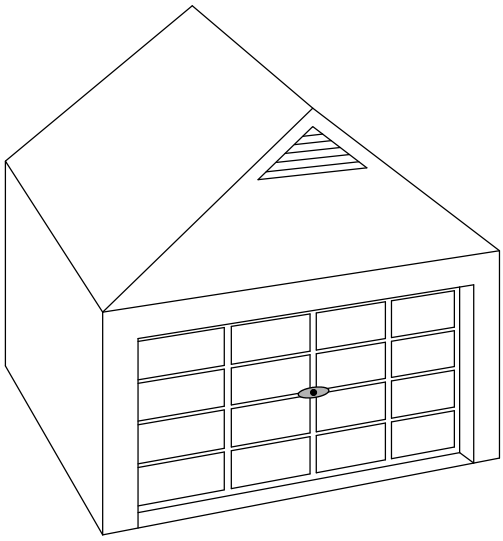
- Make each state a stage. Remember to number stages in octal. Up to 256 total stages are available in the DL105, numbered 0 to 377 in octal.
- Put transition logic inside the stage which originates each transition (the stage each arrow points *away* from).
- Use an initial stage (ISG) for any states that must be active at powerup.
- Place the outputs or actions in the appropriate stages.

You'll notice that Steps 1 through 3 just *prepare* us to write the stage program in Step 4. However, the program virtually writes itself because of the preparation beforehand. Soon you'll be able to start with a word description of an application and create a stage program in one easy session!

Stage Program Example: A Garage Door Opener

Garage Door Opener Example

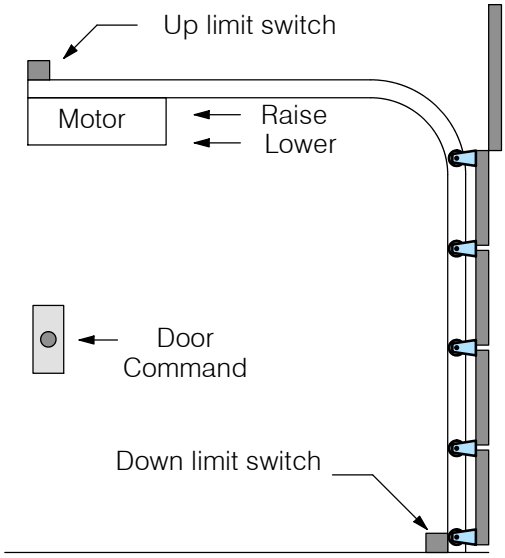
In this next stage programming example we'll create a garage door opener controller. Hopefully most readers are familiar with this application, and we can have fun besides!



The first step we must take is to describe how the door opener works. We will start by achieving the basic operation, waiting to add extra features later. Stage programs are very easy to modify.

Our garage door controller has a motor which raises or lowers the door on command. The garage owner pushes and releases a momentary pushbutton once to raise the door. After the door is up, another push-release cycle will lower the door.

In order to identify the inputs and outputs of the system, it's sometimes helpful to sketch its main components, as shown in the door side view to the right. The door has an up limit and a down limit switch. Each limit switch closes only when the door has reach the end of travel in the corresponding direction. In the middle of travel, neither limit switch is closed.

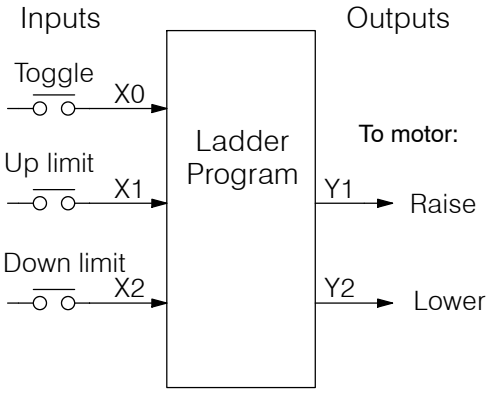


The motor has two command inputs: raise and lower. When neither input is active, the motor is stopped.

The door command is just a simple pushbutton. Whether wall-mounted as shown, or a radio-remote control, all door control commands logical OR together as one pair of switch contacts.

Draw the Block Diagram

The block diagram of the controller is shown to the right. Input X0 is from the pushbutton door control. Input X1 energizes when the door reaches the full up position. Input X2 energizes when the door reaches the full down position. When the door is positioned between fully up or down, both limit switches are open.

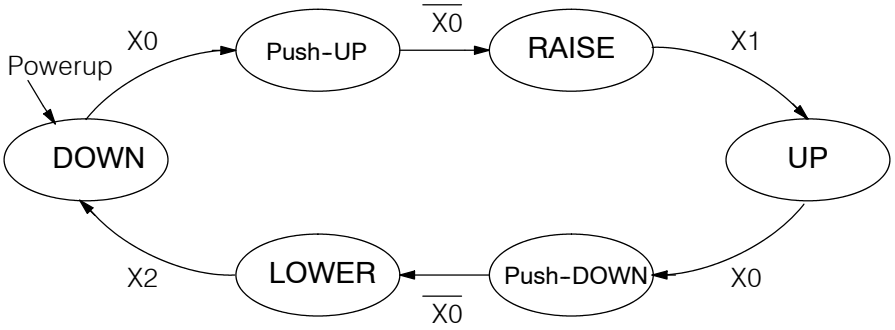


The controller has two outputs to drive the motor. Y1 is the up (raise the door) command, and Y2 is the down (lower the door) command.

Draw the State Diagram

Now we are ready to draw the state transition diagram. Like the previous light bulb controller example, this application also has just one switch for the command input. Refer to the figure below.

- When the door is down (DOWN state), nothing happens until X0 energizes. Its push and release brings us to the RAISE state, where output Y1 turns on and causes the motor to raise the door.
- We transition to the UP state when the up limit switch (X1) energizes, and turns off the motor.
- Then nothing happens until another X0 press-release cycle occurs. That takes us to the LOWER state, turning on output Y2 to command the motor to lower the door. We transition back to the DOWN state when the down limit switch (X2) energizes.

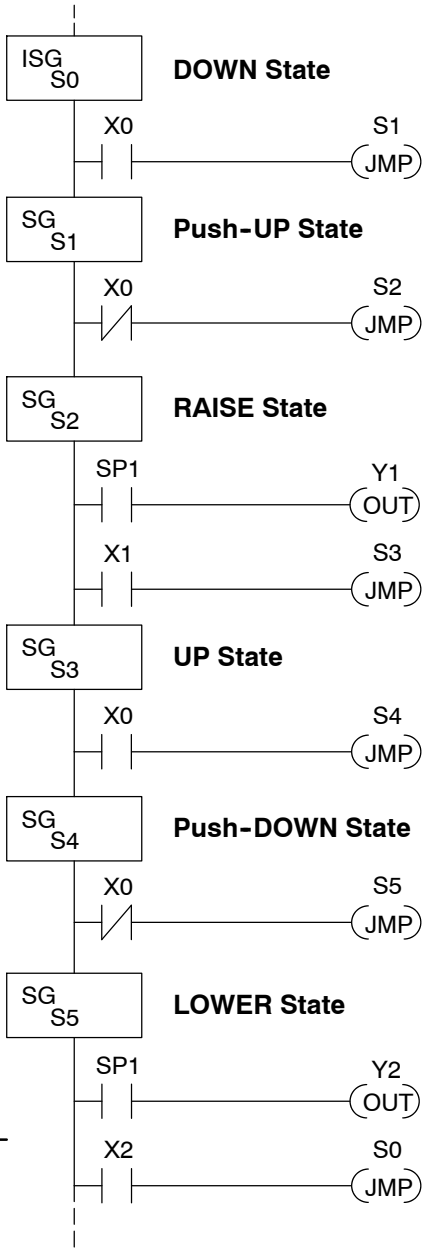


Output equations: Y1 = RAISE Y2 = LOWER

The equivalent stage program is shown to the right. For now, we will assume the door is down at powerup, so the desired powerup state is DOWN. We make S0 an initial stage (ISG). Stage S0 remains active until the door control pushbutton activates. Then we transition (JMP) to Push-UP stage, S1.

A push-release cycle of the pushbutton takes us through stage S1 to the RAISE stage, S2. We use the always-on contact SP1 to energize the motor's raise command, Y1. When the door reaches the fully-raised position, the up limit switch X1 activates. This takes us to the UP Stage S3, where we wait until another door control command occurs.

In the UP Stage S3, a push-release cycle of the pushbutton will take us to the LOWER Stage S5, where we activate Y2 to command the motor to lower the door. This continues until the door reaches the down limit switch, X2. When X2 closes, we transition from Stage S5 to the DOWN stage S0, where we began.



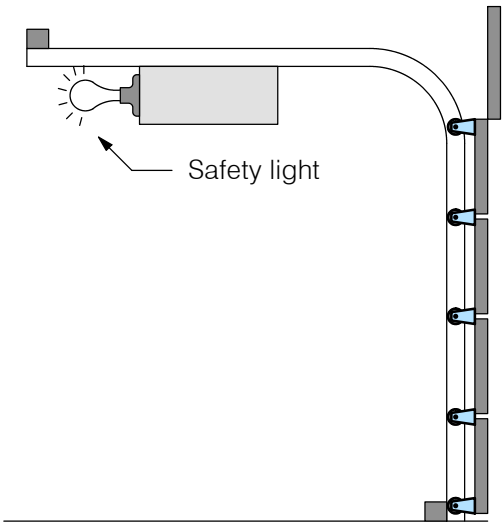
NOTE: The only special thing about an initial stage (ISG) is that it is automatically active at powerup. Afterwards, it is just like any other.

Add Safety Light Feature

Next we will add a safety light feature to the door opener system. It's best to get the main function working first as we have done, then adding the secondary features.

The safety light is standard on many commercially-available garage door openers. It is shown to the right, mounted on the motor housing. The light turns on upon any door activity, remaining on for approximately 3 minutes afterwards.

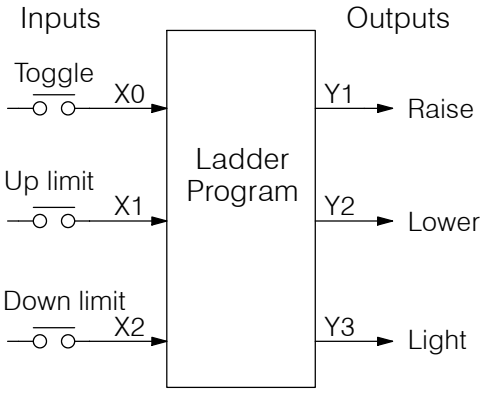
This part of the exercise will demonstrate the use of parallel states in our state diagram. Instead of using the JMP instruction, we'll use the set and reset commands.



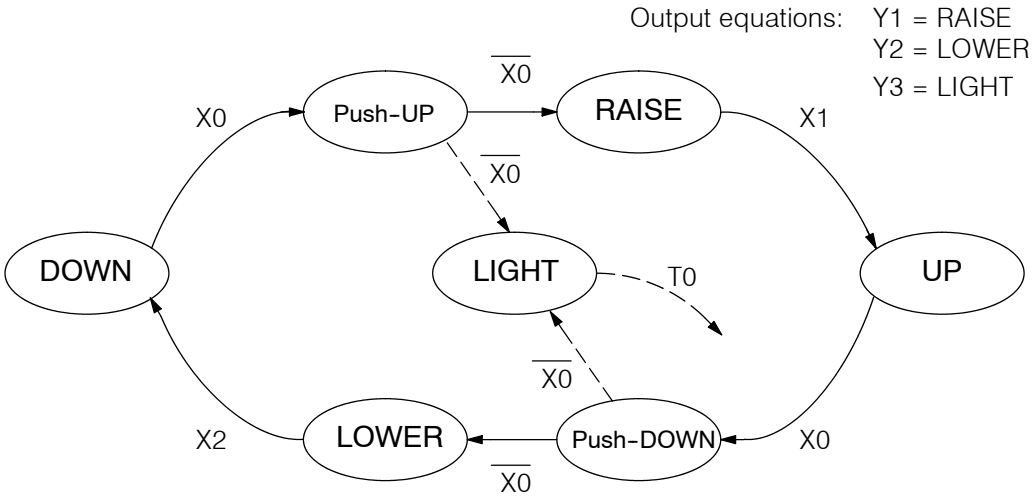
Modify the Block Diagram and State Diagram

To control the light bulb, we add an output to our controller block diagram, shown to the right, Y3 is the light control output.

In the diagram below, we add an additional state called "LIGHT". Whenever the garage owner presses the door control switch and releases, the RAISE or LOWER state is active *and the LIGHT state is simultaneously active.* The line to the Light state is dashed, because it is not the primary path.



We can think of the Light state as a parallel process to the raise and lower state. The paths to the Light state are not a transition (Stage JMP), but a State Set command. In the logic of the Light stage, we will place a three-minute timer. When it expires, timer bit T0 turns on and resets the Light stage. The path out of the Light stage goes nowhere, indicating the Light stage just becomes inactive, and the light goes out!



Using a Timer Inside a Stage

The finished modified program is shown to the right. The shaded areas indicate the program additions.

In the Push-UP stage S1, we add the Set Stage Bit S6 instruction. When contact X0 opens, we transition from S1 and go to two new active states: S2 and S6. In the Push-DOWN state S4, we make the same additions. So, any time someone presses the door control pushbutton, the light turns on.

Most new stage programmers would be concerned about where to place the Light Stage in the ladder, and how to number it. The good news is that it doesn't matter!

- Just choose an unused stage number, and use it for the new stage and as the reference from other stages.
- Placement in the program is not critical, so we place it at the end.

You might think that each stage has to be directly under the stage that transitions to it. While it is good practice, it is not required (that's good, because our two locations for the Set S6 instruction make that impossible). Stage numbers and how they are used determines the transition paths.

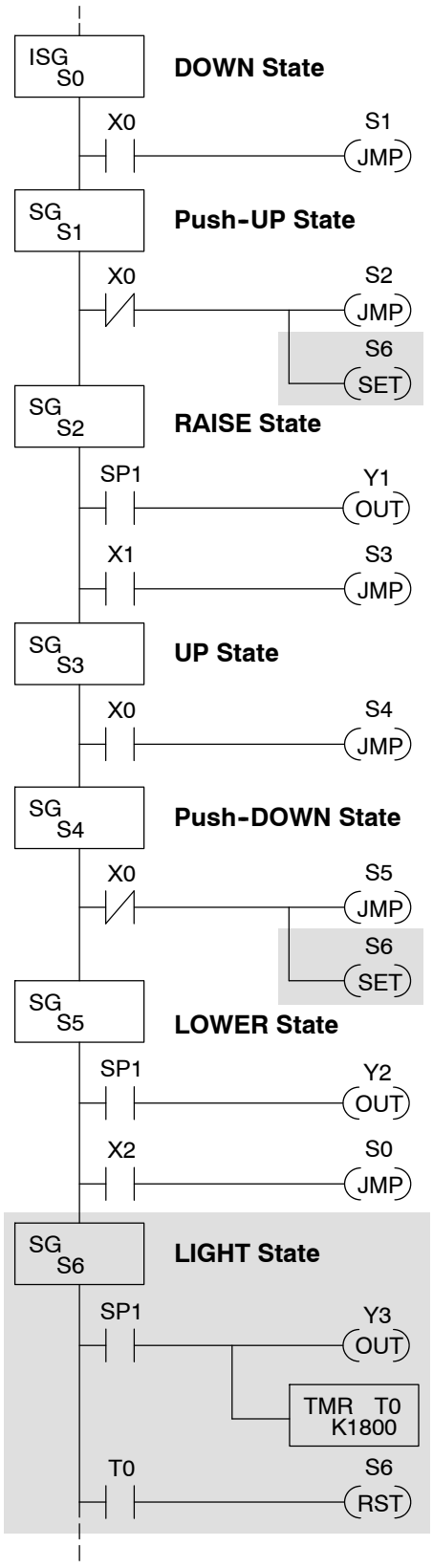
In stage S6, we turn on the safety light by energizing Y3. Special relay contact SP1 is always on. Timer T0 times at 0.1 second per count. To achieve 3 minutes time period, we calculate:

$$K = \frac{3 \text{ min.} \times 60 \text{ sec/min}}{0.1 \text{ sec/count}}$$

$$K = 1800 \text{ counts}$$

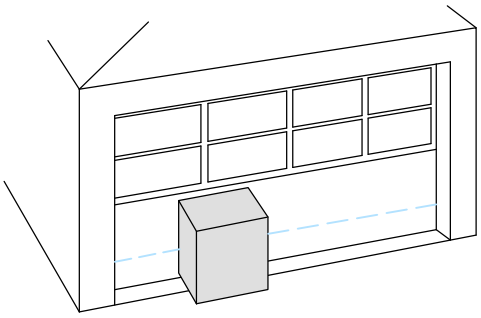
The timer has power flow whenever stage S6 is active. The corresponding timer bit T0 is set when the timer expires. So three minutes later, T0=1 and the instruction Reset S6 causes the stage to be inactive.

While Stage S6 is active and the light is on, stage transitions in the primary path continue normally and independently of Stage 6. That is, the door can go up, down, or whatever, but the light will be on for precisely 3 minutes.

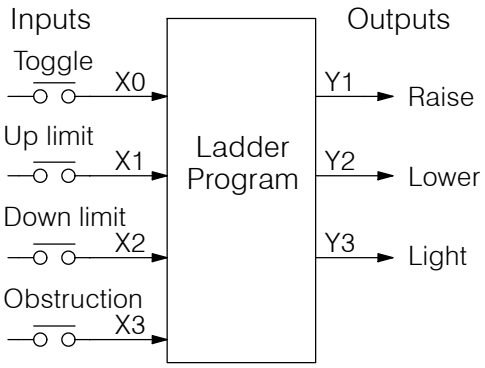


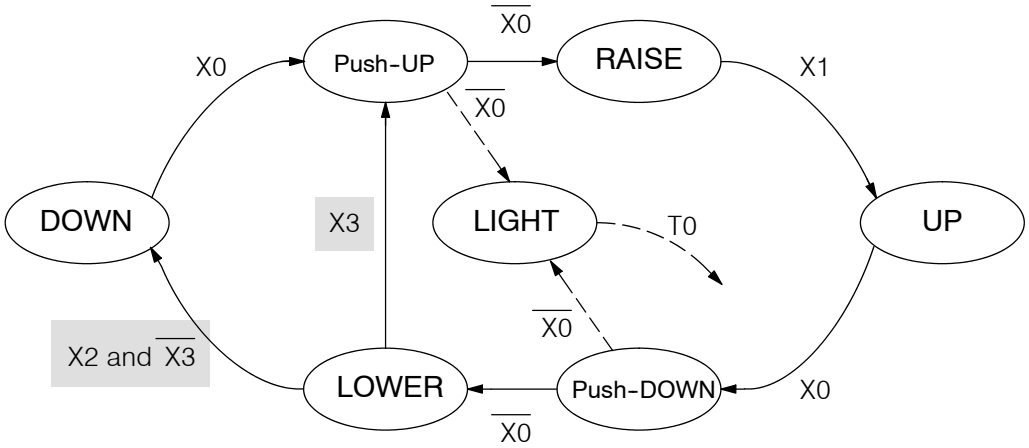
Add Emergency Stop Feature

Some garage door openers today will detect an object under the door. This halts further lowering of the door. Usually implemented with a photocell ("electric-eye"), a door in the process of being lowered will halt and begin raising. We will define our safety feature to work in this way, adding the input from the photocell to the block diagram as shown to the right. X3 will be on if an object is in the path of the door.



Next, we make a simple addition to the state transition diagram, shown in shaded areas in the figure below. Note the new transition path at the top of the LOWER state. If we are lowering the door and detect an obstruction (X3), we then jump to the Push-UP State. We do this instead of jumping directly to the RAISE state, to give the Lower output Y2 one scan to turn off, before the Raise output Y1 energizes.

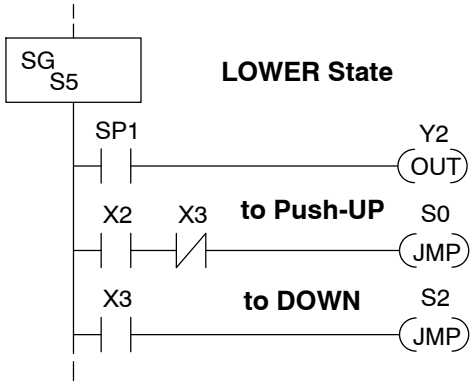




Exclusive Transitions

It is theoretically possible that the down limit (X2) and the obstruction input (X3) could energize at the same moment. In that case, we would "jump" to the Push-UP and DOWN states simultaneously, which does not make sense.

Instead, we give priority to the obstruction by changing the transition condition to the DOWN state to [X2 AND NOT X3]. This ensures the obstruction event has the priority. The modifications we must make to the LOWER stage (S5) logic are shown to the right. The first rung remains unchanged. The second and third rungs implement the transitions we need. Note the opposite relay contact usage for X3, which ensures the stage will execute only one of the JMP instructions.

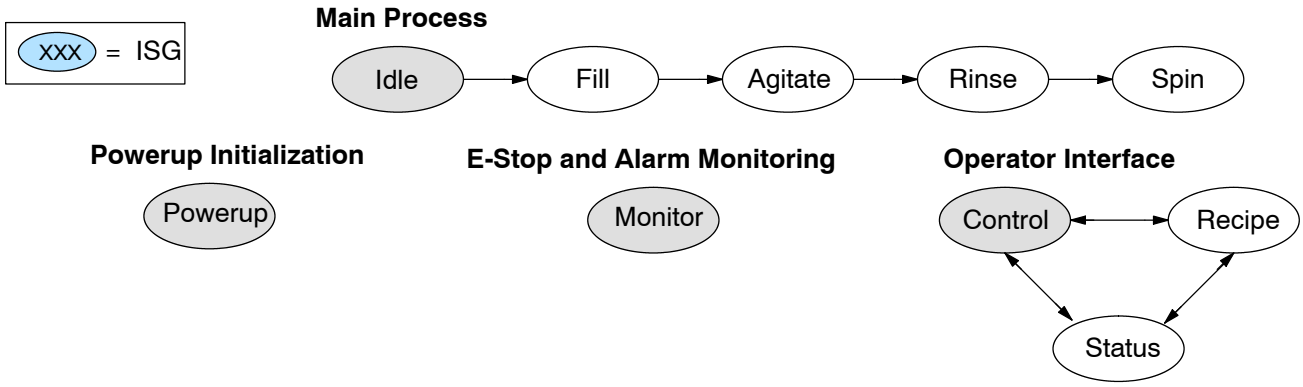


Stage Program Design Considerations

Stage Program Organization

The examples so far in this chapter used one self-contained state diagram to represent the main process. However, we can have multiple processes implemented in stages, all in the same ladder program. New stage programmers sometimes try to turn a stage on and off each scan, based on the false assumption that only one stage can be on at a time. For ladder rungs that you want to execute each scan, just put them in a stage that is always on.

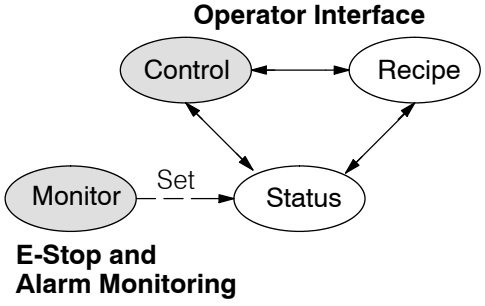
The following figure shows a typical application. During operation, the primary manufacturing activity Main Process, Powerup Initialization, E-Stop and Alarm Monitoring, and Operator Interface are all running. At powerup, three initial stages shown begin operation.



In a typical application, the separate stage sequences above operate as follows:

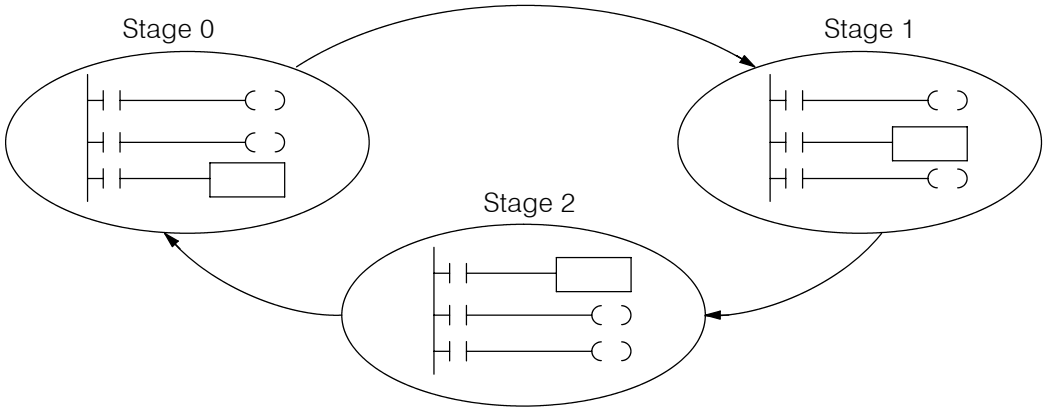
- **Powerup Initialization** – This stage contains ladder rung tasks done just once at powerup. Its last rung resets the stage, so this stage is only active for one scan (or only as many scans that are required).
- **Main Process** – this stage sequence controls the heart of the process or machine. One pass through the sequence represents one part cycle of the machine, or one batch in the process.
- **E-Stop and Alarm Monitoring** – This stage is always active because it is watching for errors that could indicate an alarm condition or require an emergency stop. It is common for this stage to reset stages in the main process or elsewhere, in order to initialize them after an error condition.
- **Operator Interface** – this is another task that must always be active and ready to respond to an operator. It allows an operator interface to change modes, etc. independently of the current main process step.

Although we have separate processes, there can be coordination among them. For example, in an error condition, the Status Stage may want to automatically switch the operator interface to the status mode to show error information as shown to the right. The monitor stage could set the stage bit for Status and Reset the stages Control and Recipe.



How Instructions Work Inside Stages

We can think of states or stages as simply dividing up our ladder program as depicted in the figure below. Each stage contains only the ladder rungs which are needed for the corresponding state of the process. The logic for transitioning out of a stage is contained within that stage. It's easy to choose which ladder rungs are active at powerup by using an "initial" stage type (ISG).



Most all instructions work just like they do in standard RLL. You can think of a stage just like a miniature RLL program which is either active or inactive.

Output Coils – As expected, output coils in active stages will turn on or off outputs according to power flow into the coil. However, note the following:

- Outputs work as usual, provided each output reference (such as "Y3") is used in only one stage.
- An output can be referenced from more than one stage, as long as only one of the stages is active at a time.
- If an output coil is controlled by more than one stage simultaneously, the active stage nearest the bottom of the program determines the final output status during each scan. Therefore, use the OROUT instruction instead when you want multiple stages to have a logical OR control of an output.

One-Shot or PD coils – Use care if you must use a Positive Differential coil in a stage. Remember that the input to the coil must make a 0–1 transition. If the coil is already energized on the first scan when the stage becomes active, the PD coil will not work. This is because the 0–1 transition did not occur.

PD coil alternative: If there is a task which you want to do only once (on 1 scan), it can be placed in a stage which transitions to the next stage on the same scan.

Counter – In using a counter inside a stage, the stage must be active for one scan before the input to the counter makes a 0–1 transition. Otherwise, there is no real transition and the counter will not count.

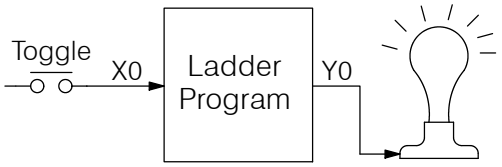
The ordinary Counter instruction does have a restriction inside stages: it may not be reset from other stages using the RST instruction for the counter bit. However, the special Stage Counter provides a solution (see next paragraph).

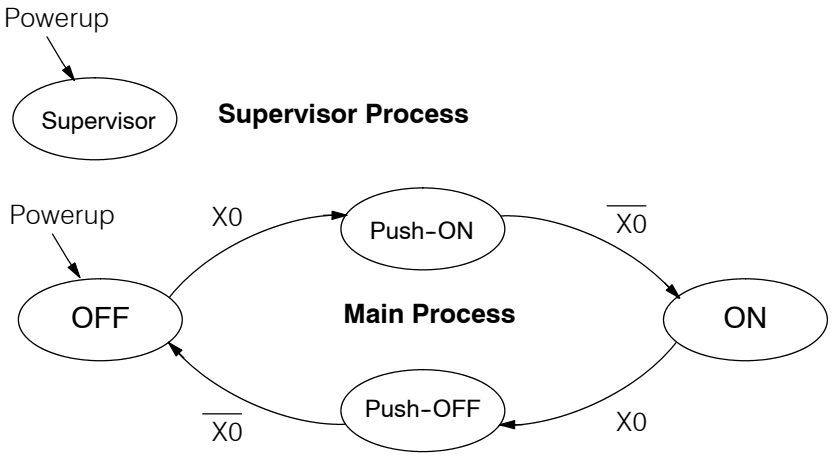
Stage Counter – The Stage Counter has the benefit that its count may be globally reset from other stages by using the RST instruction. It has a count input, but no reset input. This is the only difference from a standard counter.

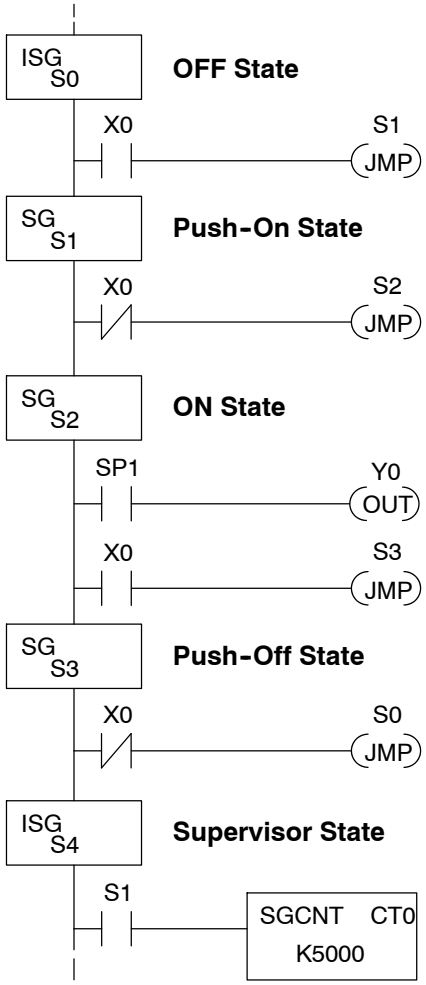
Drum – Realize that the drum sequencer is its own process, and is a different programming method than stage programming. If you need to use a drum with stages, be sure to place the drum instruction in an ISG stage that is always active.

Using a Stage as a Supervisory Process

You may recall the light bulb on-off controller example from earlier in this chapter. For the purpose of illustration, suppose we want to monitor the "productivity" of the lamp process, by counting the number of on-off cycles which occurs. This application will require the addition of a simple counter, but the key decision is in where to put the counter.







New stage programming students will typically try to place the counter inside one the the stages of the process they are trying to monitor. The problem with this approach is that the stage is active only part of the time. In order for the counter to count, the count input must transition from off to on at least one scan after its stage activates. Ensuring this requires extra logic that can be tricky.

In this case, we only need to add another supervisory stage as shown above, to "watch" the main process. The counter inside the supervisor stage uses the stage bit S1 of the main process as its count input. *Stage bits used as a contact let us monitor a process!*

Note that both the Supervisor stage and the OFF stage are initial stages. The supervisor stage remains active indefinitely.

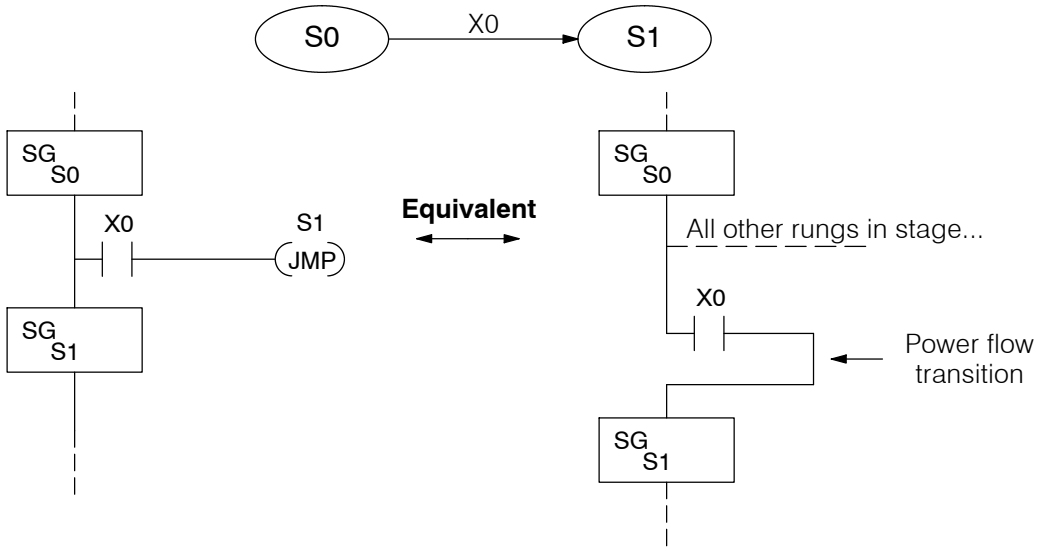
Stage Counter

The counter in the above example is a special Stage Counter. Note that it does not have a reset input. The count is reset by executing a Reset instruction, naming the counter bit (CT0 in this case). The Stage Counter has the benefit that its count may be globally reset from other stages. The standard Counter instruction does not have this global reset capability. You may still use a regular Counter instruction inside a stage... however, the reset input to the counter is the only way to reset it.

Power Flow Transition Technique

Our discussion of state transitions has shown how the stage JMP instruction makes the current stage inactive and the next stage (named in the JMP) active. As an alternative way to enter this in ***Direct***SOFT, you may use the power flow method for stage transitions.

The main requirement is that the current stage be located directly above the next (jump-to) stage in the ladder program. This arrangement is shown in the diagram below, by stages S0 and S1, respectively.

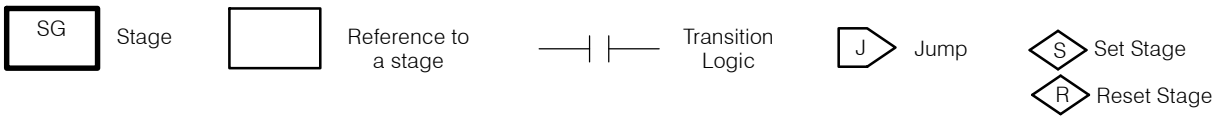


Recall that the stage JMP instruction may occur anywhere in the current stage, and the result is the same. However, power flow transitions (shown above) must occur as the last rung in a stage. All other rungs in the stage will precede it. The power flow transition method is also achievable on the handheld programmer, by simply following the transition condition with the stage instruction for the next stage.

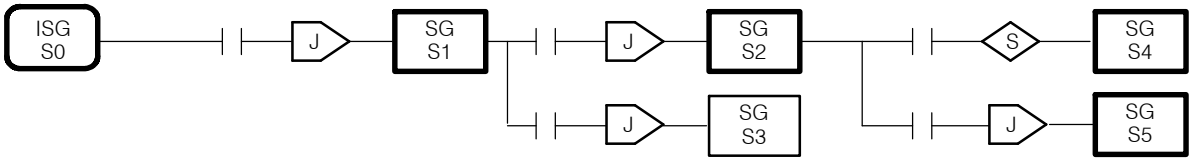
The power flow transition method does eliminate one stage JMP instruction, its only advantage. However, it is not as easy to make program changes as using the stage JMP. Therefore, we advise using stage JMP transitions for most programmers.

Stage View in *Direct*SOFT

The stage View option in ***Direct***SOFT will let you view the ladder program as a flow chart. The figure below shows the symbol convention used in the diagrams. You may find the stage view useful as a tool to verify that your stage program has faithfully reproduced the logic of the state transition diagram you intend to realize.



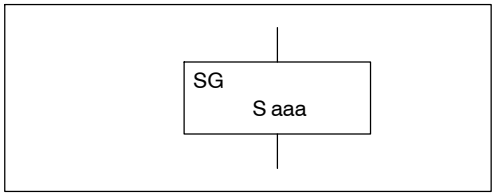
The following diagram is a typical stage view of a ladder program containing stages. Note the left-to-right direction of the flow chart.



RLL*PLUS* Stage Instructions

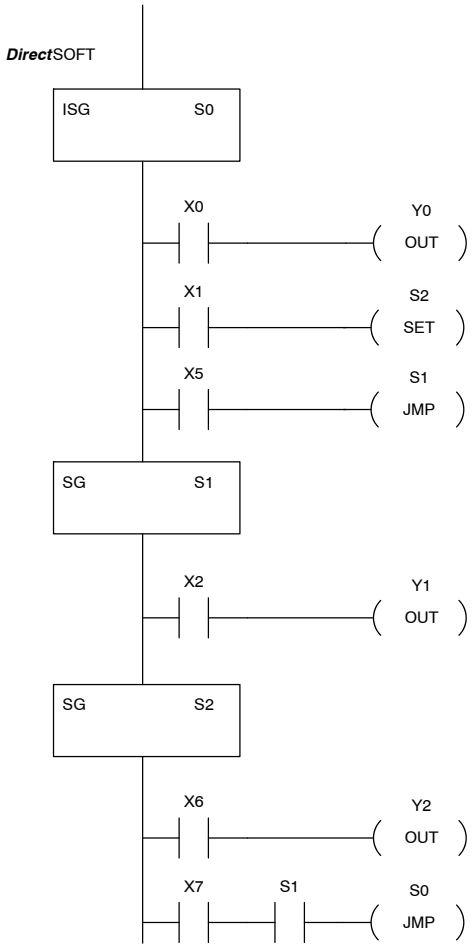
Stage (SG)

The Stage instructions are used to create structured RLL*PLUS* programs. Stages are program segments which can be activated by transitional logic, a jump or a set stage that is executed from an active stage. Stages are deactivated one scan after transitional logic, a jump, or a reset stage instruction is executed.



Operand Data Type		DL130 Range
		aaa
Stage	S	0-377

The following example is a simple RLL*PLUS* Stage program. This program utilizes an initial stage, and jump instructions to create a structured program.

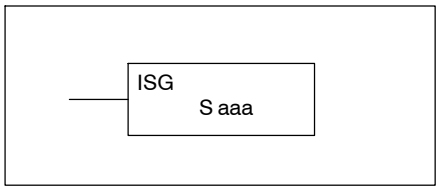


Handheld Programmer Keystrokes

U ISG	→	A 0	ENT		
$ STR	→	A 0	ENT		
GX OUT	→	A 0	ENT		
$ STR	→	B 1	ENT		
X SET	→	SHFT	S RST	C 2	ENT
$ STR	→	F 5	ENT		
K JMP	→	B 1	ENT		
2 SG	→	B 1	ENT		
$ STR	→	C 2	ENT		
GX OUT	→	B 1	ENT		
2 SG	→	C 2	ENT		
$ STR	→	G 6	ENT		
GX OUT	→	C 2	ENT		
$ STR	→	H 7	ENT		
V AND	→	SHFT	S RST	B 1	ENT
K JMP	→	A 0	ENT		

Initial Stage (ISG)

The Initial Stage instruction is normally used as the first segment of an RLL*PLUS* Stage program. Multiple Initial Stages are allowed in a program. They will be active when the CPU enters the Run mode allowing for a starting point in the program. Initial Stages are also activated by transitional logic, a jump or a set stage executed from an active stage.



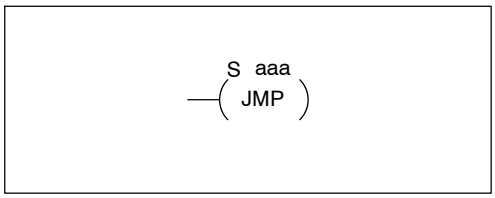
Operand Data Type		DL130 Range
		aaa
Stage	S	0-377



NOTE: If the ISG is within the retentive range for stages, the ISG will remain in the state it was in before power down and will NOT turn itself on during the first scan.

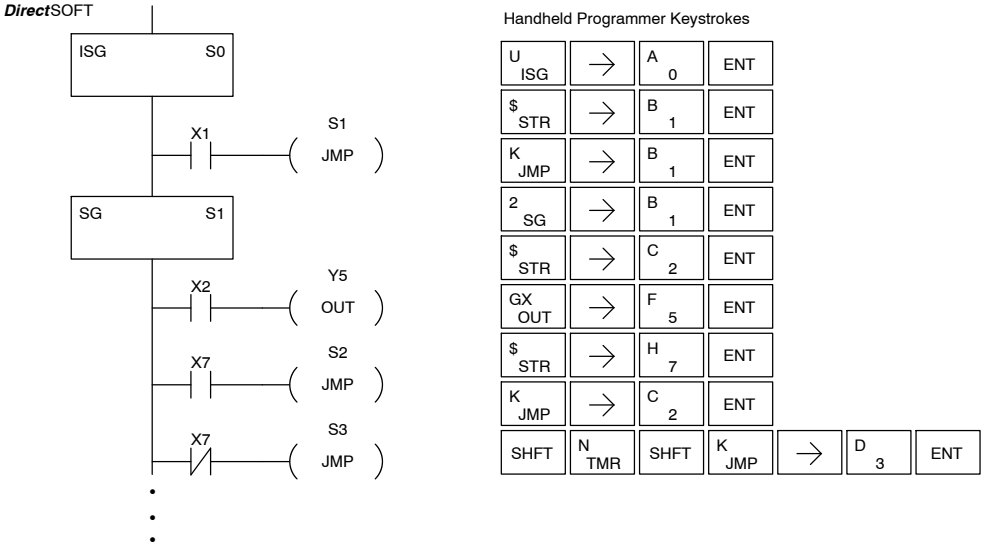
JUMP (JMP)

The Jump instruction allows the program to transition from an active stage containing the jump instruction to another stage (specified in the instruction). The jump occurs when the input logic is true. The active stage containing the Jump will deactivate 1 scan later.



Operand Data Type		DL130 Range
		aaa
Stage	S	0-377

In the following example, only stage ISG0 will be active when program execution. begins. When X1 is on, program execution will jump from Initial Stage 0 to Stage 1.





NOTE: The F1-130 CPU does not have the Not Jump instruction (as does other PLC families). You may still achieve the same result by using the Jump instruction, while inverting the sense of contact logic that activates that instruction.

Questions and Answers about Stage Programming

We include the following commonly-asked questions about Stage Programming as an aid to new students. All question topics are covered in more detail in this chapter.

Q. What does stage programming do that I can't do with regular RLL programs?

A. Stages allow you to identify all the states of your process before you begin programming. This approach is more organized, because you divide up a ladder program into sections. As stages, these program sections are active only when they are actually needed by the process. Most processes can be organized into a sequence of stages, connected by event-based transitions.

Q. Isn't a stage really just like a software subroutine?

A. No, it is very different. A subroutine is called by a main program when needed, and executes just once before returning to the point from which it was called. A stage, however, is part of the main program. It represents a state of the process, so an active stage executes on every scan of the CPU until it becomes inactive.

Q. What are Stage Bits?

A. A stage bit is just a single bit in the CPU's image register, representing the active/inactive status of the stage in real time. For example, the bit for Stage 0 is referenced as "S0". If S0 = 0, then the ladder rungs in Stage 0 are bypassed (not executed) on each CPU scan. If S0 = 1, then the ladder rungs in Stage 0 are executed on each CPU scan. Stage bits, when used as contacts, allow one part of your program to monitor another part by detecting stage active/inactive status.

Q. How does a stage become active?

A. There are three ways:

- If the Stage is an initial stage (ISG), it is automatically active at powerup.
- Another stage can execute a stage JMP instruction naming this stage, which makes it active upon its next occurrence in the program.
- A program rung can execute a Set Stage Bit instruction (such as Set S0).

Q. How does a stage become inactive?

A. There are three ways:

- Standard Stages (SG) are automatically inactive at powerup.
- A stage can execute a stage JMP instruction, resetting its Stage Bit to 0.
- Any rung in the program can execute a Reset Stage Bit instruction (such as Reset S0).

Q. What about the power flow technique of stage transitions?

A. The power flow method of connecting adjacent stages (directly above or below in the program) actually is the same as the stage Jump instruction executed in the stage above, naming the stage below. Power flow transitions are more difficult to edit in ***Direct***SOFT, we list them separately from two preceding questions.

Q. Can I have a stage which is active for only one scan?

A. Yes, but this is not the intended use for a stage. Instead, just make a ladder rung active for 1 scan by including a stage Jump instruction at the bottom of the rung. Then the ladder will execute on the last scan before its stage jumps to a new one.

Q. Isn't a stage JMP just like a regular GOTO instruction used in software?

A. No, it is very different. A GOTO instruction sends the program execution immediately to the code location named by the GOTO. A stage JMP simply resets the stage Bit of the current stage, while setting the stage Bit of the stage named in the JMP instruction. Stage bits are 0 or 1, determining the inactive/active status of the corresponding stages. A stage JMP has the following results:

- When the JMP is executed, the remainder of the current stage's rungs are executed, even if they reside past(under) the JMP instruction. On the following scan, that stage is not executed, because it is inactive.
- The stage named in the stage JMP instruction will be executed upon its next occurrence. If located past (under) the current stage, it will be executed on the same scan. If located before (above) the current stage, it will be executed on the following scan.

Q. How can I know when to use stage JMP, versus a Set Stage Bit or Reset Stage Bit?

A. These instructions are used according to the state diagram topology you have derived:

- Use a stage JMP instruction for a state transition... moving from one state to another.
- Use a Set Stage Bit instruction when the current state is spawning a new parallel state or stage sequence, or when a supervisory state is starting a state sequence under its command.
- Use a Reset Bit instruction when the current state is the last state in a sequence and its task is complete, or when a supervisory state is ending a state sequence under its command.

Q. What is an initial stage, and when do I use it?

A. An initial stage (ISG) is automatically active at powerup. Afterwards, it works just like any other stage. You can have multiple initial stages, if required. Use an initial stage for ladder that must always be active, or as a starting point.

Q. Can I have place program ladder rungs outside of the stages, so they are always on?

A. It is possible, but it's not good software design practice. Place ladder that must always be active in an initial stage, and do not reset that stage or use a stage JMP instruction inside it. It can start other stage sequences at the proper time by setting the appropriate stage Bit(s).

Q. Can I have more than one active stage at a time?

A. Yes, and this is a normal occurrence for many programs. However, it is important to organize your application into separate processes, each made up of stages. And a good process design will be mostly sequential, with only one stage on at a time. However, all the processes in the program may be active simultaneously.

8

Maintenance and Troubleshooting

In This Chapter. . . .

- Hardware System Maintenance
- Diagnostics
- CPU Indicators
- Communications Problems
- I/O Point Troubleshooting
- Noise Troubleshooting
- Machine Startup and Program Troubleshooting

Hardware System Maintenance

Standard Maintenance

No regular or preventative maintenance is required for this product (there are no internal batteries); however, a routine maintenance check (about every one or two months) of your PLC and control system is good practice, and should include the following items:

- **Air Temperature** – Check the air temperature in the control cabinet, so the operating temperature range of any component is not exceeded.
- **Air Filter** – If the control cabinet has an air filter, clean or replace it periodically as required.
- **Fuses or breakers** – verify that all fuses and breakers are intact.
- **DL105 Air Vents** – check that all air vents are clear. If the exterior case needs cleaning, disconnect the input power, and carefully wipe the case using a damp cloth. Do not let water enter the case through the air vents and do not use strong detergents because this may discolor the case.

Diagnostics

Diagnostics

Your DL105 Micro PLC performs many pre-defined diagnostic routines with every CPU scan. The diagnostics can detect various errors or failures in the PLC. The two primary error classes are *fatal* and *non-fatal*.

Fatal Errors

Fatal errors are errors which may cause the system to function improperly, perhaps introducing a safety problem. The CPU will automatically switch to Program Mode if it is in Run Mode. (Remember, in Program Mode all outputs are turned off.) If the fatal error is detected while the CPU is in Program Mode, the CPU will not allow you to transition to Run Mode until the error has been corrected.

Some examples of fatal errors are:

- Power supply failure
- Parity error or CPU malfunction
- Particular programming errors

Non-fatal Errors

Non-fatal errors are errors that need your attention, but should not cause improper operation. They do not cause or prevent any mode transitions of the CPU. The application program can use special relay contacts to detect non-fatal errors, and even take the system to an orderly shutdown or switch the CPU to Program Mode if desired. An example of a non-fatal error is:

- Particular programming errors

Finding Diagnostic Information

The programming devices will notify you of an error if one occurs while online.

- ***Direct***SOFT provides the error number and an error message.
- The handheld programmer displays error numbers and short descriptions of the error.

Appendix B has a complete list of error messages in order by error number.

Many error messages point to supplemental V-memory locations which contain related information. Special relays (SP contacts) also provide error indications.

V-memory Error Code Locations

The following table names the specific memory locations that correspond to certain types of error messages.

Error Class	Error Category	Diagnostic V-memory
User-Defined	Error code used with FAULT instruction	V7751
System Error	Fatal Error code	V7755
	Major Error code	V7756
	Minor Error code	V7757
Grammatical	Address where syntax error occurs	V7763
	Error Code found during syntax check	V7764
CPU Scan	Number of scans since last Program to Run Mode transition	V7765
	Current scan time (ms)	V7775
	Minimum scan time (ms)	V7776
	Maximum scan time (ms)	V7777

Special Relays (SP) Corresponding to Error Codes

The special relay table also includes status indicators which can indicate errors. For a more detailed description of each of these special relays refer to Appendix D.

CPU Status Relays	
SP12	Terminal Run mode
SP16	Terminal Program mode
SP20	STOP instruction was executed
SP22	Interrupt enabled
System Monitoring Relays	
SP40	Critical error
SP41	Non-critical error
SP44	Program memory error
SP50	Fault instruction was executed
SP51	Watchdog timeout
SP52	Syntax error
SP53	Cannot solve the logic

Accumulator Status Relays	
SP60	Acc. is less than value
SP61	Acc. is equal to value
SP62	Acc. is greater than value
SP63	Acc. result is zero
SP64	Half borrow occurred
SP65	Borrow occurred
SP66	Half carry occurred
SP67	Carry occurred
SP70	Result is negative (sign)
SP71	Pointer reference error
SP73	Overflow
SP75	Data is not in BCD
SP76	Load zero

DL105 Micro PLC Error Codes

These errors can be generated by the CPU or by the Handheld Programmer, depending on the actual error. Appendix B provides a more complete description of the error codes.

The errors can be detected at various times. However, most of them are detected at power-up, on entry to Run Mode, or when a Handheld Programmer key sequence results in an error or an illegal request.

Error Code	Description
E003	Software time-out
E004	Invalid instruction (RAM parity error in the CPU)
E099	Program memory exceeded
E151	Invalid command
E155	RAM failure
E210	Power fault
E312	Communications error 2
E313	Communications error 3
E316	Communications error 6
E320	Time out
E321	Communications error
E501	Bad entry
E502	Bad address
E503	Bad command
E504	Bad reference / value
E505	Invalid instruction
E506	Invalid operation
E520	Bad operation – CPU in Run
E524	Bad operation – CPU in Program

Error Code	Description
E526	Unit is offline
E527	Unit is online
E528	CPU mode
E540	CPU locked
E541	Wrong password
E542	Password reset
E601	Memory full
E602	Instruction missing
E604	Reference missing
E620	Out of memory
E621	EEPROM Memory not blank
E622	No Handheld Programmer EEPROM
E624	V memory only
E625	Program only
E627	Bad write operation
E628	Memory type error (should be EEPROM)
E640	Mis-compare
E650	Handheld Programmer system error
E651	Handheld Programmer ROM error
E652	Handheld Programmer RAM error

Program Error Codes

The following table lists program syntax and runtime error codes. Error detection occurs during a Program-to-Run mode transition, or when you use AUX 21 – Check Program. The CPU will also turn on SP52 and store the error code in V7755. Appendix B provides a more complete description of the error codes.

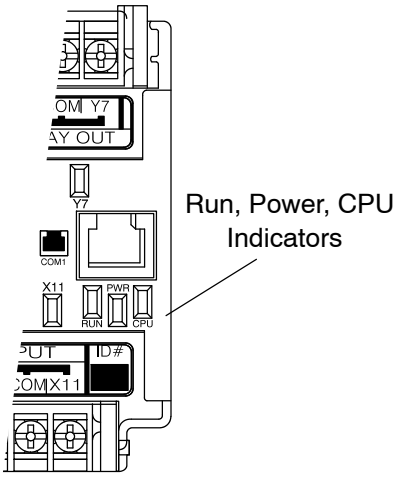
Error Code	Description
E4**	No Program in CPU
E401	Missing END statement
E402	Missing LBL
E406	Missing IRT
E421	Duplicate stage reference
E422	Duplicate SBR/LBL reference
E431	Invalid ISG/SG address
E436	Invalid INT address
E438	Invalid IRT address
E440	Invalid Data Address
E441	ACON/NCON
E451	Bad MLS/MLR

Error Code	Description
E452	X input used as output coil
E453	Missing T/C
E454	Bad TMRA
E455	Bad CNT
E456	Bad SR
E461	Stack Overflow
E462	Stack Underflow
E463	Logic Error
E464	Missing Circuit
E471	Duplicate coil reference
E472	Duplicate TMR reference
E473	Duplicate CNT reference

CPU Indicators

The DL105 Micro PLCs have indicators on the front to help you diagnose problems with the system. In normal runtime operation only the RUN and PWR indicators are on. The table below is a quick reference to potential problems.

Indicator Status	Potential Problems
PWR (LED off)	1. System voltage incorrect 2. PLC power supply faulty
RUN (LED off)	1. CPU programming error 2. (CPU in program mode)
CPU (LED on)	1. Electrical noise interference 2. Internal CPU defective

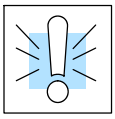


PWR Indicator

In general there are three reasons for the CPU power status LED (PWR) to be OFF:

1. Power to the unit is incorrect or is not applied.
2. PLC power supply is faulty.
3. Other component(s) have the power supply shut down.

If the voltage to the power supply is not correct, the PLC may not operate properly or may not operate at all. Use the following guidelines to correct the problem.



WARNING: To minimize the risk of electrical shock, always disconnect the system power before inspecting the physical wiring.

1. First, disconnect the external power.
2. Verify that all external circuit breakers or fuses are still intact.
3. Check all incoming wiring for loose connections. If you're using a separate termination block, check those connections for accuracy and integrity.
4. If the connections are acceptable, reconnect the system power and verify the voltage at the DL105 power input is within specification. If the voltage is not correct shut down the system and correct the problem.
5. If all wiring is connected correctly and the incoming power is within the specifications, the PLC internal supply may be faulty.

The best way to check for a faulty PLC is to substitute a known good one to see if this corrects the problem. The removable connectors on the DL105 make this relatively easy. If there has been a major power surge, it is possible the PLC internal power supply has been damaged. If you suspect this is the cause of the power supply damage, consider installing an AC line conditioner to attenuate damaging voltage spikes in the future.

RUN Indicator

If the CPU will not enter the Run mode (the RUN indicator is off), the problem is usually in the application program, unless the CPU has a fatal error. If a fatal error has occurred, the CPU LED should be on. (You can use a programming device to determine the cause of the error.)

Both of the programming devices, Handheld Programmer and ***Direct***SOFT, will return an error message describing the problem. Depending on the error, there may also be an AUX function you can use to help diagnose the problem. The most common programming error is “Missing END Statement”. All application programs require an END statement for proper termination. A complete list of error codes can be found in Appendix B.

CPU Indicator

If the CPU indicator is on, a fatal error has occurred in the CPU. Generally, this is not a programming problem but an actual hardware failure. You can power cycle the system to clear the error. If the error clears, you should monitor the system and determine what caused the problem. You will find this problem is sometimes caused by high frequency electrical noise introduced into the CPU from an outside source. Check your system grounding and install electrical noise filters if the grounding is suspected. If power cycling the system does not reset the error, or if the problem returns, you should replace the CPU.

Communications Problems

If you cannot establish communications with the CPU, check these items.

- The cable is disconnected.
- The cable has a broken wire or has been wired incorrectly.
- The cable is improperly terminated or grounded.
- The device connected is not operating at the correct baud rate (9600 baud).
- The device connected to the port is sending data incorrectly.
- A grounding difference exists between the two devices.
- Electrical noise is causing intermittent errors.
- The PLC has a bad communication port and should be replaced.

For problems in communicating with ***Direct***SOFT on a personal computer, refer to the ***Direct***SOFT manual. It includes a troubleshooting section that can help you diagnose PC problems in communications port setup, address or interrupt conflicts, etc.

I/O Point Troubleshooting

Possible Causes

If you suspect an I/O error, there are several things that could be causing the problem.

- High-Speed I/O configuration error
- A blown fuse in your machine or panel (the DL105 does not have internal I/O fuses)
- A loose terminal block
- The auxiliary 24 VDC supply has failed
- The Input or Output Circuit has failed

Some Quick Steps

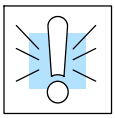
When troubleshooting the DL105 Micro PLCs there are a few facts you should be aware of. These facts may assist you in quickly correcting an I/O problem.

- HSIO configuration errors are commonly mistaken for I/O point failure during program development. If the I/O point in question is in X0–X3, or Y0–Y2, check all parameter locations listed in Chapter 3 that apply to the HSIO mode you have selected.
- The output circuits cannot detect shorted or open output points. If you suspect one or more faulty points, measure the voltage drop from the common to the suspect point. Remember when using a Digital Volt Meter, leakage current from an output device such as a triac or a transistor must be considered. A point which is off may appear to be on if no load is connected the point.
- The I/O point status indicators are logic-side indicators. This means the LED which indicates the on or off status reflects the status of the point with respect to the CPU. On an output point the status indicators could be operating normally while the actual output device (transistor, triac etc.) could be damaged. With an input point, if the indicator LED is on the input circuitry is probably operating properly. Verify the LED goes off when the input signal is removed.
- Leakage current can be a problem when connecting field devices to an I/O point. False input signals can be generated when the leakage current of an output device is great enough to turn on the connected input device. To correct this install a resistor in parallel with the input or output of the circuit. The value of this resistor will depend on the amount of leakage current and the voltage applied but usually a 10K to 20KΩ resistor will work. Verify the wattage rating of the resistor is correct for your application.
- Because of the removable terminal blocks on the DL105, the easiest method to determine if an I/O circuit has failed is to replace the unit if you have a spare. However, if you suspect a field device is defective, that device may cause the same failure in the replacement PLC as well. As a point of caution, you may want to check devices or power supplies connected to the failed I/O circuit before replacing the unit with a spare.

Testing Output Points

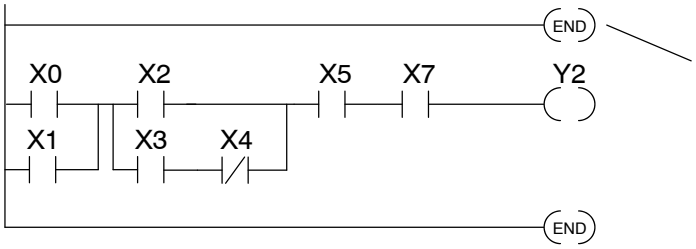
Output points can be set on or off in the DL105 series CPUs. If you want to do an I/O check out independent of the application program, follow the procedure below:

Step	Action
1	Use a handheld programmer or ***Direct***SOFT32 to communicate online to the PLC.
2	Change to Program Mode.
3	Go to address 0.
4	Insert an "END" statement at address 0. (This will cause program execution to occur only at address 0 and prevent the application program from turning the I/O points on or off).
5	Change to Run Mode.
6	Use the programming device to set (turn) on or off the points you wish to test.
7	When you finish testing I/O points delete the "END" statement at address 0.



WARNING: Depending on your application, forcing I/O points may cause unpredictable machine operation that can result in a risk of personal injury or equipment damage. Make sure you have taken all appropriate safety precautions prior to testing any I/O points.

Handheld Programmer Keystrokes Used to Test an Output Point



Insert an END statement at the beginning of the program. This disables the remainder of the program.

From a clear display, use the following keystrokes

STAT ENT

```
16P STATUS
BIT REF   X
```

Use the PREV or NEXT keys to select the Y data type

NEXT A 0 ENT

```
   Y  10    Y   0
□□□□□□□□□□□□□□□□
```

Use arrow keys to select point, then use ON and OFF to change the status

← ← SHFT ON INS

Y2 is now on

```
   Y  10    Y   0
□□□□□□□□□□□□□■□□
```

Noise Troubleshooting

Electrical Noise Problems

Noise is one of the most difficult problems to diagnose. Electrical noise can enter a system in many different ways and they fall into one of two categories, conducted or radiated. It may be difficult to determine how the noise is entering the system but the corrective actions for either of the types of noise problems are similar.

- Conducted noise is when the electrical interference is introduced into the system by way of a attached wire, panel connection ,etc. It may enter through an I/O circuit, a power supply connection, the communication ground connection, or the chassis ground connection.
- Radiated noise is when the electrical interference is introduced into the system without a direct electrical connection, much in the same manner as radio waves.

Reducing Electrical Noise

While electrical noise cannot be eliminated it can be reduced to a level that will not affect the system.

- Most noise problems result from improper grounding of the system. A good earth ground can be the single most effective way to correct noise problems. If a ground is not available, install a ground rod as close to the system as possible. Ensure all ground wires are single point grounds and are not daisy chained from one device to another. Ground metal enclosures around the system. A loose wire can act as a large antenna, introducing noise into the system. Therefore, tighten all connections in your system. Loose ground wires are more susceptible to noise than the other wires in your system. Review Chapter 2 Installation, Wiring, and Specifications if you have questions regarding how to ground your system.
- Electrical noise can enter the system through the power source for the PLC and I/O circuits. Installing an isolation transformer for all AC sources can correct this problem. DC sources should be well-grounded good quality supplies.
- Separate input wiring from output wiring. Never run low-voltage I/O wiring close to high voltage wiring.

Machine Startup and Program Troubleshooting

The DL105 Micro PLCs provide several features that can help you debug your program before and during machine startup. This section discusses the following topics which can be very helpful.

- Program Syntax Check
- Duplicate Reference Check
- Special Instructions
- Run Time Edits
- Forcing I/O Points

Syntax Check

Even though the Handheld Programmer and ***Direct***SOFT provide error checking during program entry, you may want to check a program that has been modified. Both programming devices offer a way to check the program syntax. For example, you can use AUX 21, CHECK PROGRAM to check the program syntax from a Handheld Programmer, or you can use the PLC Diagnostics menu option within ***Direct***SOFT. This check will find a wide variety of programming errors. The following example shows how to use the syntax check with a Handheld Programmer.

Use AUX 21 to perform syntax check

CLR C 2 B 1 AUX ENT

```
AUX 21 CHECK PRO
1:SYN 2:DUP REF
```

Select syntax check (default selection)

ENT (You may not get the busy display if the program is not very long.)

```
BUSY
```

One of two displays will appear

Error Display (example)

```
$00050 E401
MISSING END
```

(shows location in question)

Syntax OK display

```
NO SYNTAX ERROR
?
```

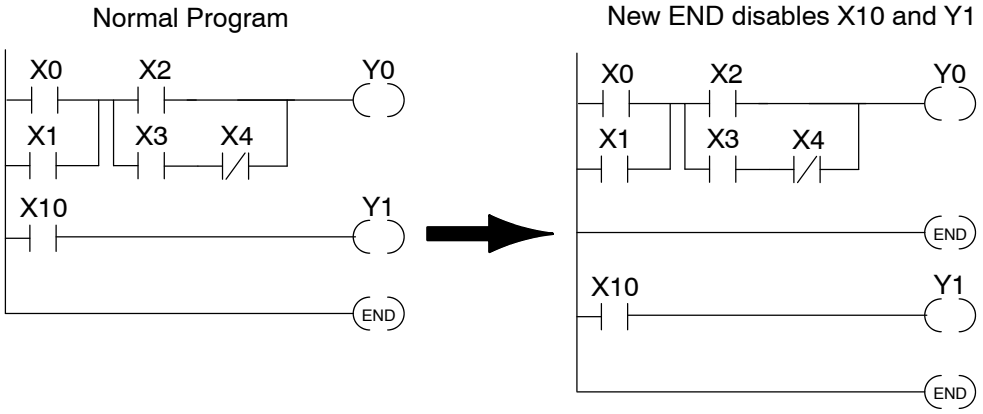
See the Error Codes Section for a complete listing of programming error codes. If you get an error, just press CLR and the Handheld will display the instruction where the error occurred. Correct the problem and continue running the Syntax check until the NO SYNTAX ERROR message appears.

Special Instructions

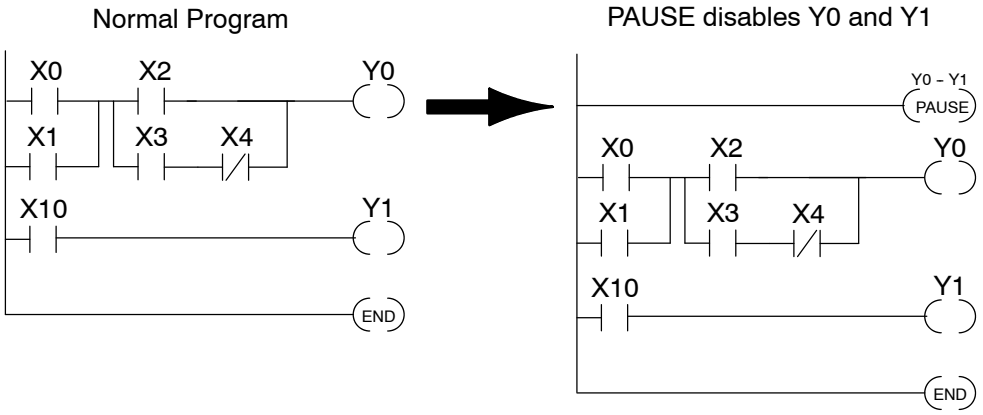
There are several instructions that can be used to help you debug your program during machine startup operations.

- END
- PAUSE
- STOP

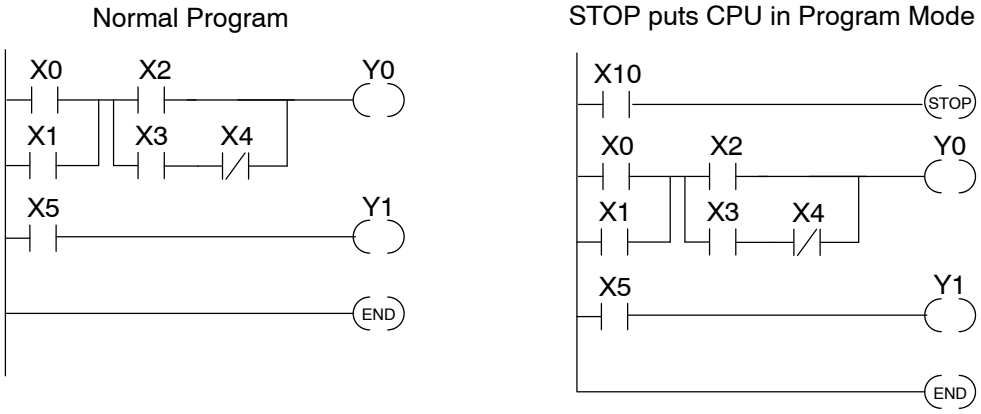
END Instruction: If you need a way to quickly disable part of the program, just insert an END statement prior to the portion that should be disabled. When the CPU encounters the END statement, it assumes that is the end of the program. The following diagram shows an example.



PAUSE Instruction: This instruction provides a quick way to allow the inputs (or other logic) to operate while disabling selected outputs. The output image register is still updated, but the output circuits are not. For example, you could make this conditional by adding an input contact or CR to control the instruction with a switch or a programming device. Or, you could just add the instruction without any conditions so the selected outputs would be disabled at all times.



STOP Instruction: Sometimes during machine startup you need a way to quickly turn off all the outputs and return to Program Mode. You can use the STOP instruction. When this instruction is executed the CPU automatically exits Run Mode and enters Program Mode. Remember, all outputs are turned off during Program Mode. The following diagram shows an example of a condition that returns the CPU to Program Mode.



In the example shown above, you could trigger X10 which would execute the STOP instruction. The CPU would enter Program Mode and all outputs would be turned off.

Duplicate Reference Check

You can also check for multiple uses of the same output coil. Both programming devices offer a way to check for this condition.. For example, you can AUX 21, CHECK PROGRAM to check for duplicate references from a Handheld Programmer, or you can use the PLC Diagnostics menu option within ***Direct***SOFT. The following example shows how to perform the duplicate reference check with a Handheld Programmer.

Use AUX 21 to perform syntax check

CLR C 2 B 1 AUX ENT

```
AUX 21 CHECK PRO
1:SYN 2:DUP REF
```

Select duplicate reference check

→ ENT (You may not get the busy display if the program is not very long.)

```
BUSY
```

One of two displays will appear

Error Display (example)
(shows location in question)

```
$00024 E471
DUP COIL REF
```

Syntax OK display

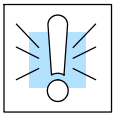
```
NO DUP REFS
?
```

If you get an error, just press CLR and the Handheld will display the instruction where the error occurred. Correct the problem and continue running the Duplicate Reference check until no duplicate references are found.

NOTE: You can use the same coil in more than one location, especially in programs containing Stage instructions and / or OROUT instructions. The Duplicate Reference check will find occurrences, even though they are acceptable.

Run Time Edits

The DL105 Micro PLC allows you to make changes to the application program during Run Mode. These edits are not "bumpless." Instead, CPU scan is momentarily interrupted (and the outputs are maintained in their current state) until the program change is complete. This means if the output is off, it will remain off until the program change is complete. If the output is on, it will remain on.



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

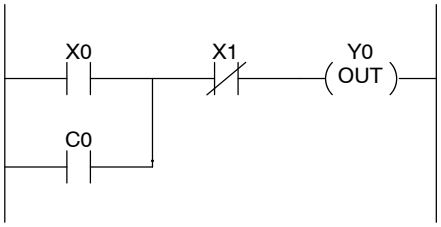
1. **If there is a syntax error in the new instruction, the CPU *will not* enter the Run Mode.**
2. **If you delete an output coil reference and the output was on at the time, the output will remain on until it is forced off with a programming device.**
3. **Input point changes are not acknowledged during Run Time Edits. So, if you're using a high-speed operation and a critical input comes on, the CPU may not see the change.**

Not all instructions can be edited during a Run Time Edit session. The following list shows the instructions that can be edited.

Mnemonic	Description
TMR	Timer
TMRF	Fast timer
TMRA	Accumulating timer
TMRAF	Accumulating fast timer
CNT	Counter
UDC	Up / Down counter
SGCNT	Stage counter
STR, STRN	Store, Store not
AND, ANDN	And, And not
OR, ORN	Or, Or not
STRE, STRNE	Store equal, Store not equal
ANDE, ANDNE	And equal, And not equal
ORE, ORNE	Or equal, Or not equal
STR, STRN	Store greater than or equal Store less than
AND, ANDN	And greater than or equal And less than

Mnemonic	Description
OR, ORN	Or greater than or equal Or less than
LD	Load data (constant)
LDD	Load data double (constant)
ADDD	Add data double (constant)
SUBD	Subtract data double (constant)
MUL	Multiply (constant)
DIV	Divide (constant)
CMPD	Compare accumulator (constant)
ANDD	And accumulator (constant)
ORD	Or accumulator (constant)
XORD	Exclusive or accumulator (constant)
LDF	Load discrete points to accumulator
OUTF	Output accumulator to discrete points
SHFR	Shift accumulator right
SHFL	Shift accumulator left
NCON	Numeric constant

We'll use the program logic shown to describe how this process works. In the example, we'll change X0 to C10. Note, the example assumes you have already placed the CPU in Run Mode.



Use the MODE key to select Run Time Edits

MODE NEXT NEXT ENT

```
*MODE CHANGE*
RUN TIME EDIT?
```

Press ENT to confirm the Run Time Edits

ENT (Note, the RUN LED on the D2-HPP Handheld starts flashing to indicate Run Time Edits are enabled.)

```
*MODE CHANGE*
RUNTIME EDITS
```

Find the instruction you want to change (X0)

SHFT X SET A 0 SHFT FD REF FIND

```
$00000 STR X0
```

Press the arrow key to move to the X. Then enter the new contact (C10).

→ → SHFT C 2 B 1 A 0 ENT

```
RUNTIME EDIT?
STR C10
```

Press ENT to confirm the change

ENT (Note, once you press ENT, the next address is displayed.

```
OR C0
```

Forcing I/O Points

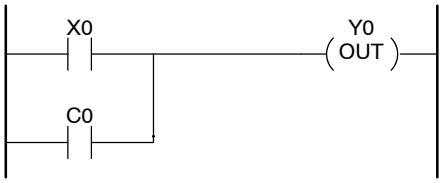
There are many times, especially during machine startup and troubleshooting, that you need the capability to force an I/O point to be either on or off. Before you use a programming device to force any data type it is important you understand how the DL105 CPUs process the forcing requests.



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

Bit Forcing — Bit forcing temporarily changes the status of a discrete bit. For example, you may want to force an input on even though the program has turned it off. This allows you to change the point status stored in the image register. The forced value will be valid until the CPU writes to the image register location during the next scan. This is useful you just need to force a bit on to trigger another event.

The following diagrams show a brief example of how you could use the D2-HPP Handheld Programmer to force an I/O point. The example assumes you have already placed the CPU into Run Mode.



From a clear display, use the following keystrokes

STAT ENT

```
16P STATUS
BIT REF   X
```

Use the PREV or NEXT keys to select the Y data type. (Once the Y appears, press 0 to start at Y0.)

NEXT A 0 ENT

```
   Y   10          Y    0
□□□□□□□□□□□□□□□□
```

Use arrow keys to select point, then use ON and OFF to change the status

← ← SHFT ON INS

Y2 is now on

```
   Y   10          Y    0
□□□□□□□□□□□■□□
```

Bit Forcing with Direct Access

From a blank display, use the following keystrokes to force Y7 ON

SHFT Y MLS H 7 SHFT ON INS

Solid fill indicates point is on.

```
BIT FORCE
Y7                    ■
```

From a blank display, use the following keystrokes to force Y7 OFF

SHFT Y MLS H 7 SHFT OFF DEL

No fill indicates point is off.

```
BIT FORCE
Y7                    □
```

Auxiliary Functions

A

In This Appendix. . . .

— Introduction
— AUX 2* — RLL Operations
— AUX 3* — V-memory Operations
— AUX 4* — I/O Configuration
— AUX 5* — CPU Configuration
— AUX 6* — Handheld Programmer Configuration
— AUX 7* — EEPROM Operations
— AUX 8* — Password Operations

Introduction

Purpose of Auxiliary Functions

Many CPU setup tasks involve the use of Auxiliary (AUX) Functions. The AUX Functions perform many different operations, including clearing ladder memory, displaying the scan time, and copying programs to EEPROM in the handheld programmer. They are divided into categories that affect different system resources. You can access the AUX Functions from ***Direct***SOFT 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 ***Direct***SOFT package. Even though this Appendix provides many examples of how the AUX functions operate, you should supplement this information with the documentation for your choice of programming device. Note, the Handheld Programmer may have additional AUX functions that are not supported with the DL105 PLCs.

AUX Function and Description		DL105
AUX 2* — RLL Operations		
21	Check Program	❍
22	Change Reference	❍
23	Clear Ladder Range	❍
24	Clear All Ladders	❍
AUX 3* — V-Memory Operations		
31	Clear V Memory	❍
AUX 4* — I/O Configuration		
41	Show I/O Configuration	❍
AUX 5* — CPU Configuration		
51	Modify Program Name	❍
53	Display Scan Time	❍
54	Initialize Scratchpad	❍
55	Set Watchdog Timer	❍
57	Set Retentive Ranges	❍
58	Test Operations / Pause Override	❍
5B	HSIO Interface Configuration	❍

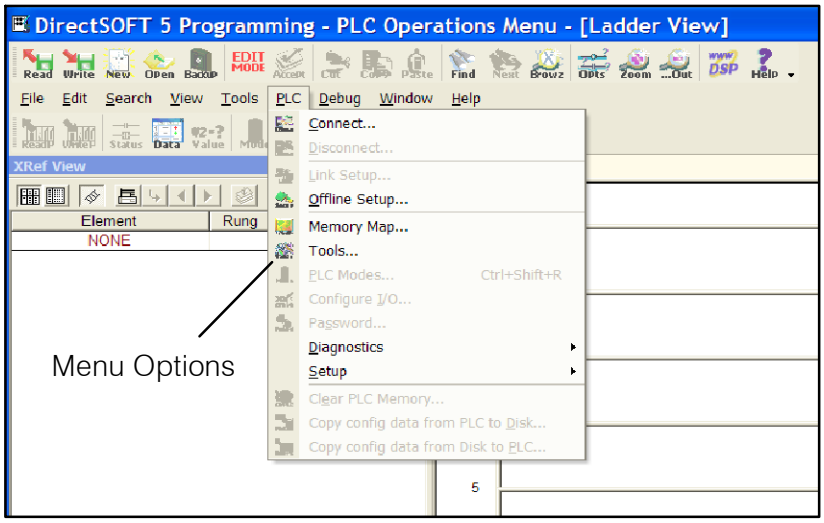
AUX Function and Description		DL105
AUX 6* — Handheld Programmer Configuration		
61	Show Revision Numbers	❍
62	Beeper On / Off	HP
65	Run Self Diagnostics	HP
AUX 7* — EEPROM Operations		
71	Copy CPU memory to HPP EEPROM	HP
72	Write HPP EEPROM to CPU	HP
73	Compare CPU to HPP EEPROM	HP
74	Blank Check (HPP EEPROM)	HP
75	Erase HPP EEPROM	HP
76	Show EEPROM Type (CPU and HPP)	HP
AUX 8* — Password Operations		
81	Modify Password	❍
82	Unlock CPU	❍
83	Lock CPU	❍

❍ — supported

HP — Handheld Programmer function

Accessing AUX Functions via *Direct*SOFT

DirectSOFT provides various menu options during both online and offline programming. Some of the AUX functions are only available during online programming, some only during offline programming, and some during both online and offline programming. The following diagram shows and example of the PLC operations menu available within ***Direct***SOFT.



Accessing AUX Functions via the Handheld Programmer

You can also access the AUX functions by using a Handheld Programmer. Plus, remember some of the AUX functions are only available from the Handheld. Sometimes the AUX name or description cannot fit on one display. If you want to see the complete description, just press the arrow keys to scroll left and right. Also, depending on the current display, you may have to press CLR more than once.

CLR AUX

```
AUX FUNCTION SELECTION
AUX 2* RLL OPERATIONS
```

Use NXT or PREV to cycle through the menus

NEXT

```
AUX FUNCTION SELECTION
AUX 3* V OPERATIONS
```

Press ENT to select sub-menus

ENT

```
AUX 3* V OPERATIONS
AUX 31 CLR V MEMORY
```

You can also enter the exact AUX number to go straight to the sub-menu.

Enter the AUX number directly

CLR D 3 B 1 AUX

```
AUX 3* V OPERATIONS
AUX 31 CLR V MEMORY
```

AUX 2* — RLL Operations

RLL Operations auxiliary functions allow you to perform various operations on the ladder program.

AUX 21
Check Program

Both the Handheld and ***Direct***SOFT automatically check for errors during program entry. However, there may be occasions when you want to check a program that has already been in the CPU. Two types of checks are available:

- Syntax
- Duplicate References

The Syntax check will find a wide variety of programming errors, such as missing END statements. If you perform this check and get an error, see Appendix B for a complete listing of programming error codes. Correct the problem and then continue running the Syntax check until the message "NO SYNTAX ERROR appears.

Use the Duplicate Reference check to verify you have not used the same output coil reference more than once. Note, this AUX function will also find the same outputs even if they have been used with the OROUT instruction, which is perfectly acceptable.

This AUX function is available on the PLC Diagnostics sub-menu from within ***Direct***SOFT.

AUX 22
Change Reference

There will probably be times when you need to change an I/O address reference or control relay reference. AUX 22 allows you to quickly and easily change all occurrences, (within an address range), of a specific instruction. For example, you can replace every instance of X5 with X10.

AUX 23
Clear Ladder Range

There have been many times when we've taken existing programs and added or removed certain portions to solve new application problems. By using AUX 23 you can select and delete a portion of the program. ***Direct***SOFT does not have a menu option for this AUX function, but you can just select the appropriate portion of the program and cut it with the editing tools.

AUX 24
Clear Ladders

AUX 24 clears the entire program from CPU memory. Before you enter a new program, you should always clear ladder memory. This AUX function is available on the PLC/Clear PLC sub-menu within ***Direct***SOFT.

AUX 3* — V-memory Operations

AUX 31
Clear V-Memory

AUX 31 clears all the information from the V-memory locations available for general use. This AUX function is available on the PLC/Clear PLC sub-menu within ***Direct***SOFT.

AUX 4* — I/O Configuration

AUX 41
Show I/O Configuration

This AUX function allows you to display the current I/O configuration on the DL105. Both the Handheld Programmer and ***Direct***SOFT will show the I/O configuration is fixed at F1-130:10 DI / 8 DO.

AUX 5* — CPU Configuration

The following auxiliary AUX functions allow you to setup, view, or change the CPU configuration.

AUX 51
Modify Program Name

DL105 PLCs can use a program name for the CPU program or a program stored on EEPROM in the Handheld Programmer. (Note, you cannot have multiple programs stored on the EEPROM.) The program name can be up to eight characters in length and can use any of the available characters (A–Z, 0–9). AUX 51 allows you to enter a program name. You can also perform this operation from within ***Direct***SOFT by using the PLC/Setup sub-menu. Once you've entered a program name, you can only clear the name by using AUX 54 to reset the system memory. Make sure you understand the possible effects of AUX 54 before you use it!

AUX 53
Display Scan Time

AUX 53 displays the current, minimum, and maximum scan times. The minimum and maximum times are the ones that have occurred since the last Program Mode to Run Mode transition. You can also perform this operation from within ***Direct***SOFT by using the PLC/Diagnostics sub-menu.

AUX 54
Initialize Scratchpad

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



NOTE: You may never have to use this feature unless you have made changes that affect system memory. Usually, you'll only need to initialize the system memory if you are changing programs and the old program required a special system setup. You can usually change from program to program without ever initializing system memory.

AUX 54 resets the system memory to the default values. You can also perform this operation from within ***Direct***SOFT by using the PLC/Setup sub-menu.

AUX 55
Set Watchdog Timer

DL105 PLCs have a "watchdog" timer that is used to monitor the scan time. The default value set from the factory is 200 ms. If the scan time exceeds the watchdog time limit, the CPU automatically leaves RUN mode and enters PGM mode. The Handheld displays the following message E003 S/W TIMEOUT when the scan overrun occurs.

Use AUX 55 to increase or decrease the watchdog timer value. You can also perform this operation from within ***Direct***SOFT by using the PLC/Setup sub-menu.

AUX 57
Set Retentive Ranges

DL105 CPUs provide certain ranges of retentive memory by default. Some of the retentive memory locations are backed up by a super-capacitor, and others are in non-volatile EEPROM. The EEPROM memory locations are V4000 to 4177, and V7630 to V7647. 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 (see Appendix E).

The default settings are:

Memory Area	DL105	
	Default Range	Available Range
Control Relays	C300 – C377	C0 – C377
V Memory	V2000 – V7777	V0 – V7777
Timers	None by default	T0 – T77
Counters	CT0 – CT77	CT0 – CT77
Stages	None by default	S0 – S377

Use AUX 57 to change the retentive ranges. You can also perform this operation from within ***Direct***SOFT by using the PLC/Setup sub-menu.



WARNING: The DL105 CPUs do not have battery-backed RAM. The super-capacitor will retain the values in the event of a power loss, but only up to 3 weeks. (The retention time may be as short as 4 1/2 days in 60 degree C operating temperature.)

AUX 58 Test Operations

AUX 58 is used to override the output disable function of the Pause instruction. Use AUX 58 to program a single output or a range of outputs which will operate normally even when those points are within the scope of the pause instruction.

AUX 5B Counter Interface Configuration

AUX 5B is used with the High-Speed I/O (HSIO) function to select the configuration. You can choose the type of counter, set the counter parameters, etc. See Chapter 3 for a complete description of how to select the various counter features.

AUX 6* — Handheld Programmer Configuration

The following auxiliary functions allow you to setup, view, or change the Handheld Programmer configuration.

AUX 61 Show Revision Numbers

As with most industrial control products, there are cases when additional features and enhancements are made. Sometimes these new features only work with certain releases of firmware. By using AUX 61 you can quickly view the CPU and Handheld Programmer firmware revision numbers. This information (for the CPU) is also available from within ***Direct***SOFT from the PLC/Diagnostics sub-menu.

AUX 62 Beeper On / Off

The Handheld has a beeper that provides confirmation of keystrokes. You can use Auxiliary (AUX) Function 62 to turn off the beeper.

AUX 65 Run Self Diagnostics

If you think the Handheld Programmer is not operating correctly, you can use AUX 65 to run a self diagnostics program. You can check the following items.

- Keypad
- Display
- LEDs and Backlight
- Handheld Programmer EEPROM check

AUX 7* — EEPROM Operations

The following auxiliary functions allow you to move the ladder program from one area to another and perform other program maintenance tasks.

Transferrable Memory Areas

Many of these AUX functions allow you to copy different areas of memory to and from the CPU and handheld programmer. The following table shows the areas that may be mentioned.

Option and Memory Type	DL105 Default Range
1:PGM — Program	$00000 – $02047
2:V — V–memory	$00000 – $03777
3:SYS — System	Non-selectable copies system parameters
4:etc (All)— Program, System and *non-volatile* V–memory only	Non-selectable

AUX 71 CPU to HPP EEPROM

AUX 71 copies information from the CPU memory to an EEPROM installed in the Handheld Programmer.You can copy different portions of EEPROM (HP) memory to the CPU memory as shown in the previous table.

AUX 72 HPP EEPROM to CPU

AUX 72 copies information from the EEPROM installed in the Handheld Programmer to CPU memory in the DL105. You can copy different portions of EEPROM (HP) memory to the CPU memory as shown in the previous table.

AUX 73 Compare HPP EEPROM to CPU

AUX 73 compares the program in the Handheld programmer (EEPROM) with the CPU program. You can compare different types of information as shown previously.

AUX 74 HPP EEPROM Blank Check

AUX 74 allows you to check the EEPROM in the handheld programmer to make sure it is blank. It's a good idea to use this function anytime you start to copy an entire program to an EEPROM in the handheld programmer.

AUX 75 Erase HPP EEPROM

AUX 75 allows you to clear all data in the EEPROM in the handheld programmer. You should use this AUX function before you copy a program from the CPU.

AUX 76 Show EEPROM Type

You can use AUX 76 to quickly determine what size EEPROM is installed in the Handheld Programmer.

AUX 8* — Password Operations

There are several AUX functions available that you can use to modify or enable the CPU password. You can use these features during on-line communications with the CPU, or, you can also use them with an EEPROM installed in the Handheld Programmer during off-line operation. This will allow you to develop a program in the Handheld Programmer and include password protection.

- AUX 81 — Modify Password
- AUX 82 — Unlock CPU
- AUX 83 — Lock CPU

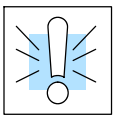
AUX 81
Modify Password

You can use AUX 81 to provide an extra measure of protection by entering a password that prevents unauthorized machine operations. The password must be an eight-character numeric (0–9) code. Once you've entered a password, you can remove it by entering all zeros (00000000). (This is the default from the factory.)

Once you've entered a password, you can lock the CPU against access. There are two ways to lock the CPU with the Handheld Programmer.

- The CPU is always locked after a power cycle (if a password is present).
- You can use AUX 82 and AUX 83 to lock and unlock the CPU.

You can also enter or modify a password from within ***Direct***SOFT by using the PLC/Password sub-menu. This feature works slightly differently in ***Direct***SOFT. Once you've entered a password, the CPU is automatically locked when you exit the software package. It will also be locked if the CPU is power cycled.



WARNING: Make *sure* you remember the password *before* you lock the CPU. Once the CPU is locked you cannot view, change, or erase the password. If you do not remember the password, you have to return the CPU to the factory for password removal.



NOTE: The DL105 CPUs support multi-level password protection of the ladder program. This allows password protection while not locking the communication port to an operator interface. The multi-level password can be invoked by creating a password with an upper case "A" followed by seven numeric characters (e.g. A1234567).

AUX 82
Unlock CPU

AUX 82 can be used to unlock a CPU that has been password protected. ***Direct***SOFT will automatically ask you to enter the password if you attempt to communicate with a CPU that contains a password.

AUX 83
Lock CPU

AUX 83 can be used to lock a CPU that contains a password. Once the CPU is locked, you will have to enter a password to gain access. Remember, this is not necessary with ***Direct***SOFT since the CPU is automatically locked whenever you exit the software package.

DL105 Error Codes

B

In This Appendix. . . .

— Error Code Table

DL105 Error Code	Description
E003 SOFTWARE TIME-OUT	If the program scan time exceeds the time allotted to the watchdog timer, this error will occur. SP51 will be on and the error code will be stored in V7755. To correct this problem use AUX 55 to extend the time allotted to the watchdog timer.
E004 INVALID INSTRUCTION	The CPU attempted to execute an instruction code, but the RAM contents had a parity error. Performing a program download to the CPU in an electrically noisy environment can corrupt a program's contents. Clear the CPU program memory, and download the program again.
E099 PROGRAM MEMORY EXCEEDED	If the compiled program length exceeds the amount of available CPU RAM this error will occur. SP52 will be on and the error code will be stored in V7755. Reduce the size of the application program.
E151 BAD COMMAND	A parity error has occurred in the application program. SP44 will be on and the error code will be stored in V7755 .This problem may possibly be due to electrical noise. Clear the memory and download the program again. Correct any grounding problems. If the error returns replace the Micro PLC.
E155 RAM FAILURE	A checksum error has occurred in the system RAM. SP44 will be on and the error code will be stored in V7755. This problem may possibly be due to a low battery, electrical noise or a CPU RAM failure. Clear the memory and download the program again. Correct any grounding problems. If the error returns replace the CPU.
E210 POWER FAULT	A short duration power drop-out occurred on the main power line supplying power to the base.

DL105 Error Code	Description
E312 HP COMM ERROR 2	A data error was encountered during communications with the CPU. Clear the error and retry the request. If the error continues check the cabling between the two devices, replace the handheld programmer, then if necessary replace the CPU. The error code will be stored in V7756.
E313 HP COMM ERROR 3	An address error was encountered during communications with the CPU. Clear the error and retry the request. If the error continues check the cabling between the two devices, replace the handheld programmer, then if necessary replace the CPU. The error code will be stored in V7756.
E316 HP COMM ERROR 6	A mode error was encountered during communications with the CPU. Clear the error and retry the request. If the error continues replace the handheld programmer, then if necessary replace the CPU. The error code will be stored in V7756.
E320 HP COMM TIME-OUT	The CPU did not respond to the handheld programmer communication request. Check to insure cabling is correct and not defective. Power cycle the system if the error continues replace the CPU first and then the handheld programmer if necessary.
E321 COMM ERROR	A data error was encountered during communication with the CPU. Check to insure cabling is correct and not defective. Power cycle the system and if the error continues replace the CPU first and then the handheld programmer if necessary.
E4** NO PROGRAM	A syntax error exists in the application program. The most common is a missing END statement. Run AUX21 to determine which one of the E4** series of errors is being flagged. SP52 will be on and the error code will be stored in V7755.
E401 MISSING END STATEMENT	All application programs must terminate with an END statement. Enter the END statement in appropriate location in your program. SP52 will be on and the error code will be stored in V7755.
E402 MISSING LBL	A MOVMC or LDLBL instruction was used without the appropriate label. Refer to the Chapter 5 for details on these instructions. SP52 will be on and the error code will be stored in V7755.
E406 MISSING IRT	An interrupt routine in the program does not end with the IRT instruction. SP52 will be on and the error code will be stored in V7755.
E421 DUPLICATE STAGE REFERENCE	Two or more SG or ISG labels exist in the application program with the same number. A unique number must be allowed for each Stage and Initial Stage. SP52 will be on and the error code will be stored in V7755.
E422 DUPLICATE LBL REFERENCE	Two or more LBL instructions exist in the application program with the same number. A unique number must be allowed for each and label. SP52 will be on and the error code will be stored in V7755.
E431 INVALID ISG/SG ADDRESS	An ISG or SG instruction must not be placed after the end statement (such as inside a subroutine). SP52 will be on and the error code will be stored in V7755.

DL105 Error Code	Description
E436 INVALID INT ADDRESS	An INT must be programmed after the end statement, not in the main body of the program. SP52 will be on and the error code will be stored in V7755.
E438 INVALID IRT ADDRESS	An IRT must be programmed after the end statement, not in the main body of the program. SP52 will be on and the error code will be stored in V7755.
E440 INVALID DATA ADDRESS	Either the DLBL instruction has been programmed in the main program area (not after the END statement), or the DLBL instruction is on a rung containing input contact(s).
E441 ACON/NCON	An ACON or NCON must be programmed after the end statement, not in the main body of the program. SP52 will be on and the error code will be stored in V7755.
E451 BAD MLS/MLR	MLS instructions must be numbered in ascending order from top to bottom.
E452 X AS COIL	An X data type is being used as a coil output.
E453 MISSING T/C	A timer or counter contact is being used where the associated timer or counter does not exist.
E454 BAD TMRA	One of the contacts is missing from a TMRA instruction.
E455 BAD CNT	One of the contacts is missing from a CNT or UDC instruction.
E456 BAD SR	One of the contacts is missing from the SR instruction.
E461 STACK OVERFLOW	More than nine levels of logic have been stored on the stack. Check the use of OR STR and AND STR instructions.
E462 STACK UNDERFLOW	An unmatched number of logic levels have been stored on the stack. Insure the number of AND STR and OR STR instructions match the number of STR instructions.
E463 LOGIC ERROR	A STR instruction was not used to begin a rung of ladder logic.
E464 MISSING CKT	A rung of ladder logic is not terminated properly.
E471 DUPLICATE COIL REFERENCE	Two or more OUT instructions reference the same I/O point.
E472 DUPLICATE TMR REFERENCE	Two or more TMR instructions reference the same number.
E473 DUPLICATE CNT REFERENCE	Two or more CNT instructions reference the same number.

DL105 Error Code	Description
E501 BAD ENTRY	An invalid keystroke or series of keystrokes was entered into the handheld programmer.
E502 BAD ADDRESS	An invalid or out of range address was entered into the handheld programmer.
E503 BAD COMMAND	An invalid command was entered in the handheld programmer.
E504 BAD REF/VAL	An invalid value or reference number was entered with an instruction.
E505 INVALID INSTRUCTION	An invalid instruction was entered into the handheld programmer.
E506 INVALID OPERATION	An invalid operation was attempted by the handheld programmer.
E520 BAD OP-RUN	An operation which is invalid in the RUN mode was attempted by the handheld programmer.
E524 BAD OP-PGM	An operation which is invalid in the PROGRAM mode was attempted by the handheld programmer.
E526 OFF LINE	The handheld programmer is in the OFFLINE mode. To change to the ONLINE mode use the MODE the key.
E527 ON LINE	The handheld programmer is in the ON LINE mode. To change to the OFF LINE mode use the MODE the key.
E528 CPU MODE	The operation attempted is not allowed during a Run Time Edit.
E540 CPU LOCKED	The CPU has been password locked. To unlock the CPU use AUX82 with the password.
E541 WRONG PASSWORD	The password used to unlock the CPU with AUX82 was incorrect.
E542 PASSWORD RESET	The CPU powered up with an invalid password and reset the password to 00000000. A password may be re-entered using AUX81.

DL105 Error Code	Description
E601 MEMORY FULL	Attempted to enter an instruction which required more memory than is available in the CPU.
E602 INSTRUCTION MISSING	A search function was performed and the instruction was not found.
E604 REFERENCE MISSING	A search function was performed and the reference was not found.
E620 OUT OF MEMORY	An attempt to transfer more data between the CPU and handheld programmer than the receiving device can hold.
E621 EEPROM NOT BLANK	An attempt to write to a non-blank EEPROM in the handheld programmer was made. Erase the EEPROM and then retry the write.
E622 NO HPP EEPROM	A data transfer was attempted with no EEPROM (or possibly a faulty EEPROM) installed in the handheld programmer.
E623 SYSTEM EEPROM	A function was requested with an EEPROM in the handheld programmer which contains system information only.
E624 V-MEMORY ONLY	A function was requested with an EEPROM in the handheld programmer which contains V-memory data only.
E625 PROGRAM ONLY	A function was requested with an EEPROM in the handheld programmer which contains program data only.
E627 BAD WRITE	An attempt to write to a write-protected or faulty EEPROM in the handheld programmer was made. Check the write protect jumper and replace the EEPROM if necessary.
E628 EEPROM TYPE ERROR	The wrong size EEPROM is being used in the handheld programmer. This error occurs when the program size is larger than what the HPP can hold.
E640 COMPARE ERROR	A compare between the EEPROM handheld programmer and the CPU was found to be in error.
E650 HPP SYSTEM ERROR	A system error has occurred in the handheld programmer. Power cycle the handheld programmer. If the error returns replace the handheld programmer.
E651 HPP ROM ERROR	A ROM error has occurred in the handheld programmer. Power cycle the handheld programmer. If the error returns replace the handheld programmer.
E652 HPP RAM ERROR	A RAM error has occurred in the handheld programmer. Power cycle the handheld programmer. If the error returns replace the handheld programmer.

C

Instruction Execution Times

In This Appendix. . . .

— Introduction

— Instruction Execution Times

Introduction

This appendix contains several tables that provide the instruction execution times for DL105 Micro PLCs. Many of the execution times depend on the type of data used with the instruction. Registers may be classified into the following types:

- Data (word) Registers
- Bit Registers

V-Memory Data Registers

Some V-memory locations are considered data registers, such as timer or counter current values. Standard user V-memory is classified as a V-memory data register. Note that you can load a bit pattern into these types of registers, even though their primary use is for data registers. The following locations are data registers:

Data Registers	DL105
Timer Current Values	V0 – V77
Counter Current Values	V1000 – V1077
User Data Words	V2000 – V2377 V4000 – V4177

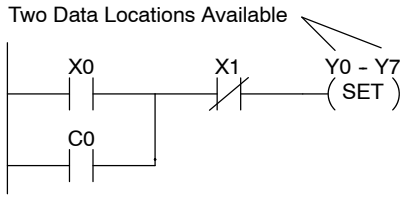
V-Memory Bit Registers

You may recall that some of the discrete points such as X, Y, C, etc. are automatically mapped into V-memory (see Appendix E). The following bit registers contain this data:

Bit Registers	DL105
Input Points (X)	V40400 – V 40407
Output Points (Y)	V40500 – V40507
Control Relays (C)	V40600 – V40617
Timer Status Bits	V41100 – V41103
Counter Status Bits	V41140 – V41143
Stages	V41000 – V41017

How to Read the Tables

Some instructions can have more than one parameter. For example, the SET instruction shown in the ladder program to the right can set a single point or a range of points.



In these cases, execution times that depend on the amount and type of parameters. The execution time tables list execution times for both situations, as shown below:

SET	1st #:	X, Y, C, S		17.4 µs
	2nd #:	X, Y, C, S,	(N pt)	12.0µs+5.4µsxN
RST	1st #:	X, Y, C, S		19.5 µs
	2nd #:	X, Y, C, S,	(N pt)	10.5µs+5.2µsxN

Execution depends on numbers of locations and types of data used

Instruction Execution Times

Boolean Instructions

Boolean Instructions		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
STR	X, Y, C, T, CT,S, SP	3.3 µs	3.3 µs
STRN	X, Y, C, T, CT,S, SP	3.9 µs	3.9 µs
OR	X, Y, C, T, CT, S, SP	2.7 µs	2.7 µs
ORN	X, Y, C, T, CT,S, SP	3.3 µs	3.3 µs
AND	X, Y, C, T, CT, S, SP	2.1 µs	2.1 µs
ANDN	X, Y, C, T, CT, S, SP	2.7 µs	2.7 µs
ANDSTR	None	1.2 µs	1.2 µs
ORSTR	None	1.2 µs	1.2 µs
OUT	X, Y, C	3.4 µs	3.4 µs
OROUT	X, Y, C	8.6 µs	8.6 µs
PD	X, Y, C	13.5 µs	13.5 µs
SET	1st #: X, Y, C, S 2nd #: X, Y, C, S (N pt)	17.4 µs 12.0µs+5.4µsxN	6.8 µs 6.8 µs
RST	1st #: T, CT 2nd #: T, CT (N pt)	31.6 µs 17µs+14.6µsxN	6.8 µs 6.8 µs
PAUSE	1wd: Y 2wd: Y (N points)	19.0 µs 15µs+4µs x N	19.0 µs 15µs+4µs x N
RST	1st #: X, Y, C, S 2nd #: X, Y, C, S (N pt)	17.7 µs 10.5µs+5.2µsxN	6.8 µs 6.8 µs

Comparative Boolean Instructions

Comparative Boolean Instructions			DL105	
Instruction	**Legal Data Types**		**Execute**	**Not Execute**
STRE	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	77 µs 158 µs 57 µs	13.8 µs 13.8 µs 13.8 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	158 µs 240 µs 139 µs	13.8 µs 13.8 µs 13.8 µs
STRNE	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	77 µs 158 µs 57 µs	13.8 µs 13.8 µs 13.8 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	158 µs 240 µs 139 µs	13.8 µs 13.8 µs 13.8 µs

Comparative Boolean (cont.)			DL105	
Instruction	**Legal Data Types**		**Execute**	**Not Execute**
ORE	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	75 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	158 µs 239 µs 137 µs	12.0 µs 12.0 µs 12.0 µs
ORNE	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	75 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	158 µs 239 µs 137 µs	12.0 µs 12.0 µs 12.0 µs
ANDE	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	75 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	158 µs 239 µs 137 µs	12.0 µs 12.0 µs 12.0 µs
ANDNE	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	75 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	158 µs 239 µs 137 µs	12.0 µs 12.0 µs 12.0 µs
STR	1st	2nd		
	T, CT	V:Data Reg. V:Bit Reg. K:Constant	78 µs 158 µs 57 µs	13.8 µs 13.8 µs 13.8 µs
	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	78 µs 159 µs 57 µs	13.8 µs 13.8 µs 13.8 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	159 µs 241 µs 139 µs	13.8 µs 13.8 µs 13.8 µs
STRN	1st	2nd		
	T, CT	V:Data Reg. V:Bit Reg. K:Constant	78 µs 158 µs 57 µs	13.8 µs 13.8 µs 13.8 µs
	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	78 µs 159 µs 57 µs	13.8 µs 13.8 µs 13.8 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	159 µs 241 µs 139 µs	13.8 µs 13.8 µs 13.8 µs

Comparative Boolean (cont.)			DL105	
Instruction	**Legal Data Types**		**Execute**	**Not Execute**
OR	1st	2nd		
	T, CT	V:Data Reg. V:Bit Reg. K:Constant	75 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	75 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	158 µs 240 µs 137 µs	12.0 µs 12.0 µs 12.0 µs
ORN	1st	2nd		
	T, CT	V:Data Reg. V:Bit Reg. K:Constant	75 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	75 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	158 µs 240 µs 137 µs	12.0 µs 12.0 µs 12.0 µs
AND	1st	2nd		
	T, CT	V:Data Reg. V:Bit Reg. K:Constant	76 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	75 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	158 µs 240 µs 137 µs	12.0 µs 12.0 µs 12.0 µs
ANDN	1st	2nd		
	T, CT	V:Data Reg. V:Bit Reg. K:Constant	76 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	1st	2nd		
	V: Data Reg.	V:Data Reg. V:Bit Reg. K:Constant	76 µs 158 µs 55 µs	12.0 µs 12.0 µs 12.0 µs
	V: Bit Reg.	V:Data Reg. V:Bit Reg. K:Constant	158 µs 240 µs 137 µs	12.0 µs 12.0 µs 12.0 µs

Immediate Instructions

Immediate Instructions		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
STRI	X	27 µs	9.8 µs
STRNI	X	26 µs	8.6 µs
ORI	X	27 µs	9.8 µs
ORNI	X	26 µs	8.6 µs
ANDI	X	25 µs	8.0 µs
ANDNI	X	24 µs	6.8 µs
OROUTI	Y	45 µs	45 µs
SETI	1st #: Y	25.5 µs	6.8 µs
	2nd #: Y (N pt)	5.5 µs+20 x N	6.8 µs
RSTI	1st #: Y	25.5 µs	6.8 µs
	2nd #: Y (N pt)	5 µs+20.5 x N	6.8 µs

Timer, Counter, Shift Register, EDRUM Instructions

Timer, Counter, Shift Register, and Drum Instructions			DL105	
Instruction	**Legal Data Types**		**Execute**	**Not Execute**
TMR	1st	2nd		
	T	V:Data Reg. V:Bit Reg. K:Constant	75 µs 158 µs 66 µs	31 µs 31 µs 31 µs
TMRF	1st	2nd		
	T	V:Data Reg. V:Bit Reg. K:Constant	75 µs 158 µs 66 µs	31 µs 31 µs 31 µs
TMRA	1st	2nd		
	T	V:Data Reg. V:Bit Reg. K:Constant	94 µs 304 µs 95 µs	56 µs 264 µs 45 µs
TMRAF	1st	2nd		
	T	V:Data Reg. V:Bit Reg. K:Constant	98 µs 304 µs 95 µs	54 µs 264 µs 49 µs
CNT	1st	2nd		
	CT	V:Data Reg. V:Bit Reg. K:Constant	68 µs 148 µs 56 µs	61 µs 141 µs 45 µs
SGCNT	1st	2nd		
	CT	V:Data Reg. V:Bit Reg. K:Constant	57 µs 140 µs 46 µs	64 µs 148 µs 53 µs

Timer, Counter, Shift Register, and Drum Instructions Cont'd		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
UDC	1st 2nd		
	CT V:Data Reg. V:Bit Reg. K:Constant	103 µs 310 µs 102 µs	74 µs 281 µs 70 µs
SR	C (N points to shift)	30 µs+4.6 µs xN	17.2 µs
EDRUM	CT	320 µs	221 µs

Accumulator Data Instructions

Accumulator / Stack Load and Output Data Instructions		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
LD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	68 µs 149 µs 62 µs 169 µs 256 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs 8.4 µs
LDD	V:Data Reg. V:Bit Reg. K:Constant P:Indir. (Data) P:Indir. (Bit)	72 µs 266 µs 64 µs 172 µs 373 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs 8.4 µs
LDF	1st 2nd X, Y, C, S K:Constant T, CT, SP (N pt)	77 µs+6.2 µs xN	10 µs
LDA	O: (Octal constant for address)	58 µs	8.4 µs
OUT	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	60 µs 132 µs 162 µs 239 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs
OUTD	V:Data Reg. V:Bit Reg. P:Indir. (Data) P:Indir. (Bit)	68 µs 276 µs 196 µs 384 µs	8.4 µs 8.4 µs 8.4 µs 8.4 µs
OUTF	1st 2nd X, Y, C K:Constant (N pt)	36 µs+7.6 µs xN	8 µs
POP	None	55 µs	7.2 µs

Logical Instructions

Logical (Accumulator) Instructions		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
AND	V:Data Reg. V:Bit Reg.	63 µs 261 µs	10.4 µs 10.4 µs
ANDD	K:Constant	53 µs	8.4 µs
OR	V:Data Reg. V:Bit Reg.	59 µs 257 µs	10.4 µs 10.4 µs
ORD	K:Constant	49 µs	8.4 µs
XOR	V:Data Reg. V:Bit Reg.	60 µs 257 µs	10.4 µs 10.4 µs
XORD	K:Constant	49 µs	8.4 µs
CMP	V:Data Reg. V:Bit Reg.	59 µs 259 µs	10.4 µs 10.4 µs
CMPD	V:Data Reg. V:Bit Reg. K:Constant	63 µs 257 µs 54 µs	8.4 µs 8.4 µs 8.4 µs

Math Instructions

Math Instructions (Accumulator)		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
ADD	V:Data Reg. V:Bit Reg.	198 µs 397 µs	10.6 µs 10.6 µs
ADDD	V:Data Reg. V:Bit Reg. K:Constant	198 µs 397 µs 188 µs	8.4 µs 8.4 µs 8.4 µs
SUB	V:Data Reg. V:Bit Reg.	200 µs 397 µs	10.6 µs 10.6 µs
SUBD	V:Data Reg. V:Bit Reg. K:Constant	198 µs 392 µs 190 µs	8.4 µs 8.4 µs 8.4 µs
MUL	V:Data Reg. V:Bit Reg. K:Constant	497 µs 483 µs 487 µs	10.6 µs 10.6 µs 8.4 µs
DIV	V:Data Reg. V:Bit Reg. K:Constant	909 µs 1108 µs 899 µs	10.6 µs 10.6 µs 8.4 µs
INCB	V:Data Reg. V:Bit Reg.	83 µs 349 µs	10.4 µs 10.4 µs
DECB	V:Data Reg. V:Bit Reg.	82 µs 351 µs	10.4 µs 10.4 µs

Bit Instructions

Bit Instructions (Accumulator)		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
SHFR	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	44μs+14.6 x N 243μs+14.6 x N 34μs+14.6 x N	10.4 μs 10.4 μs 8.4 μs
SHFL	V:Data Reg. (N bits) V:Bit Reg. (N bits) K:Constant (N bits)	44μs+14.6 x N 243μs+14.6 x N 34μs+14.6 x N	10.4 μs 10.4 μs 8.4 μs
ENCO	None	62 μs	7.2 μs
DECO	None	34 μs	7.2 μs

Number Conversion Instructions

Number Conversion Instructions (Accumulator)		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
BIN	None	359 μs	7.2 μs
BCD	None	403 μs	7.2 μs
INV	None	27 μs	5.0 μs

Table Instructions

Table Instructions		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
MOV	Move V:data reg. to V:data reg. Move V:bit reg. to V:data reg. Move V:data reg to V:bit reg. Move V:bit reg. to V:bit reg. N= #of words	450μs+17 x N 430μs+244 x N 460μs+215 x N 490μs+448 x N	6.2μs 6.2μs 6.2μs 6.2μs
MOVMC	Move V:Data Reg. to E^2 Move V:Bit Reg. to E^2 Move from E^2 to V:Data Reg. Move from E^2 to V:Bit Reg. N= #of words	— — 250μs+201xN —	— — 6.2μs —
LDLBL	K	58μs	8.4μs

CPU Control Instructions

CPU Control Instructions		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
NOP	None	0 µs	0 µs
END	None	27 µs	27 µs
STOP	None	16 µs	5 µs

Program Control Instructions

Program Control Instructions		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
MLS	K (1-7)	12 µs	12 µs
MLR	K (0-6) N= 1 to 7	13 µs + 2.4 x N	13 µs + 2.4 x N

Interrupt Instructions

Interrupt Instructions		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
ENI	None	9 µs	5 µs
DISI	None	8 µs	5 µs
INT	O0	0 µs	0 µs
IRT	None	1.6 µs	—

Message Instructions

Message Instructions		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
FAULT	V:Data Reg. V:Bit Reg. K:Constant	171 µs 253 µs 2798 µs	8.4 µs 8.4 µs 8.4 µs
DLBL	K	0 µs	0 µs
NCON	K	0 µs	0 µs
ACON	K	0 µs	0 µs

RLL*PLUS* Instructions

RLL*PLUS* Instructions		DL105	
Instruction	**Legal Data Types**	**Execute**	**Not Execute**
ISG	S	31 µs	32 µs
SG	S	31 µs	32 µs
JMP	S	14 µs	8 µs

Special Relays

D

In This Appendix. . . .

— DL105 PLC Special Relays

DL105 PLC Special Relays

"Special Relays" are just contacts that are set by the CPU operating system to indicate a particular system event has occurred. These contacts are available for use in your ladder program. Knowing just the right special relay contact to use for a particular situation can save lot of programming time. Since the CPU operating system sets and clears special relay contacts, the ladder program only has to use them as inputs in ladder logic.

Startup and Real-Time Relays

SP0	First scan	on for the first scan after a power cycle or program to run transition only. The relay is reset to off on the second scan. It is useful where a function needs to be performed only on program startup.
SP1	Always ON	provides a contact to insure an instruction is executed every scan.
SP3	1 minute clock	on for 30 seconds and off for 30 seconds.
SP4	1 second clock	on for 0.5 second and off for 0.5 second.
SP5	100 ms clock	on for 50 ms. and off for 50 ms.
SP6	50 ms clock	on for 25 ms. and off for 25 ms.
SP7	Alternate scan	on every other scan.

CPU Status Relays

SP12	Terminal run mode	on when the CPU is in the run mode.
SP16	Terminal program mode	on when the CPU is in the program mode.
SP20	Forced stop mode	on when the STOP instruction is executed.
SP22	Interrupt enabled	on when interrupts have been enabled using the ENI instruction.

System Monitoring

SP40	Critical error	on when a critical error such as I/O communication loss has occurred.
SP41	Warning	on when a non critical error such as a low battery has occurred.
SP44	Program memory error	on when a memory error such as a memory parity error has occurred.
SP50	Fault instruction	on when a Fault Instruction is executed.
SP51	Watch Dog timeout	on if the CPU Watch Dog timer times out.
SP52	Grammatical error	on if a grammatical error has occurred either while the CPU is running or if the syntax check is run. V7755 will hold the exact error code.
SP53	Solve logic error	on if CPU cannot solve the logic.

Accumulator Status

SP60	Value less than	on when the accumulator value is less than the instruction value.
SP61	Value equal to	on when the accumulator value is equal to the instruction value.
SP62	Greater than	on when the accumulator value is greater than the instruction value.
SP63	Zero	on when the result of the instruction is zero (in the accumulator.)
SP64	Half borrow	on when the 16 bit subtraction instruction results in a borrow.
SP65	Borrow	on when the 32 bit subtraction instruction results in a borrow.
SP66	Half carry	on when the 16 bit addition instruction results in a carry.
SP67	Carry	when the 32 bit addition instruction results in a carry.
SP70	Sign	on anytime the value in the accumulator is negative.
SP71	Invalid octal number	on when an Invalid octal number was entered. This also occurs when the V-memory specified by a pointer (P) is not valid.
SP73	Overflow	on if overflow occurs in the accumulator when a signed addition or subtraction results in an incorrect sign bit.
SP75	Data error	on if a BCD number is expected and a non-BCD number is encountered.
SP76	Load zero	on when any instruction loads a value of zero into the accumulator.

HSIO Pulse Catch Relay

SP100	X0 is on	X0 — on for 1 scan after a pulse on X0 occurs.

Equal Relays for HSIO Mode 10 Counter Presets

SP540	Current = target value	on when the counter current value equals the value in V2320.
SP541	Current = target value	on when the counter current value equals the value in V2322.
SP542	Current = target value	on when the counter current value equals the value in V2324.
SP543	Current = target value	on when the counter current value equals the value in V2326.
SP544	Current = target value	on when the counter current value equals the value in V2330.
SP545	Current = target value	on when the counter current value equals the value in V2332.
SP546	Current = target value	on when the counter current value equals the value in V2334.
SP547	Current = target value	on when the counter current value equals the value in V2336.
SP550	Current = target value	on when the counter current value equals the value in V2340.
SP551	Current = target value	on when the counter current value equals the value in V2342.
SP552	Current = target value	on when the counter current value equals the value in V2344.
SP553	Current = target value	on when the counter current value equals the value in V2346
SP554	Current = target value	on when the counter current value equals the value in V2350.
SP555	Current = target value	on when the counter current value equals the value in V2352.
SP556	Current = target value	on when the counter current value equals the value in V2354.
SP557	Current = target value	on when the counter current value equals the value in V2356.
SP560	Current = target value	on when the counter current value equals the value in V2360.
SP561	Current = target value	on when the counter current value equals the value in V2362.
SP562	Current = target value	on when the counter current value equals the value in V2364.
SP563	Current = target value	on when the counter current value equals the value in V2366.
SP564	Current = target value	on when the counter current value equals the value in V2370.
SP565	Current = target value	on when the counter current value equals the value in V2372.
SP566	Current = target value	on when the counter current value equals the value in V2374.
SP567	Current = target value	on when the counter current value equals the value in V2376.

PLC Memory

E

In This Appendix. . . .

— DL105 PLC Memory

DL105 PLC Memory

When designing a PLC application, it is important for the PLC user to understand the different types of memory in the PLC. Two types of memory are used by the DL105 CPU, RAM and EEPROM. This memory can be configured by the PLC user as either retentive or non-retentive memory.

Retentive memory is memory that is configured by the user to maintain values through a power cycle or a PROGRAM to RUN transition. Non-retentive memory is memory that is configured by the PLC user to clear data after a power cycle or a PROGRAM to RUN transition. The retentive ranges can be configured with either the handheld programmer using AUX 57 or ***Direct***SOFT (PLC Setup).

The contents of RAM memory can be written to and read from for an infinite number of times, but RAM requires a power source to maintain the contents of memory.The contents of RAM are maintained by the internal power supply (5VDC) only while the PLC is powered by an external source, normally 120VAC. When power to the PLC is turned off, the contents of RAM are maintained by a "Super-Capacitor". If the Super-Capacitor ever discharges, the contents of RAM will be lost. The data retention time of the Super-Capacitor backed RAM is 3 weeks maximum, and 4 1/2 days minimum (at 60° C).

The contents of EEPROM memory can be read from for an infinite number of times but there is a limit to the number of times it can be written to (typical specification is 100,000 writes). EEPROM does not require a power source to maintain the memory contents. It will retain the contents of memory indefinately.

PLC user V-memory is stored in both volatile RAM and non-volatile EEPROM memory. Data being stored in RAM uses V2000-V2377. Data stored in EEPROM uses V4000-V4177 and V7630-V7647.

Data values that must be retained for long periods of time, when the PLC is powered off, should be stored in EEPROM based V-memory.

Data values that are continually changing or which can be initialized with program logic should be stored in RAM based V-memory.

F

European Union Directives (CE)

In This Appendix. . . .

— European Union (EU) Directives
— Basic EMC Installation Guidelines

European Union (EU) Directives



NOTE: The information contained in this section is intended as a guideline and is based on our interpretation of the various standards and requirements. Since the actual standards are issued by other parties and in some cases Governmental agencies, the requirements can change over time without advance warning or notice. Changes or additions to the standards can possibly invalidate any part of the information provided in this section.

This area of certification and approval is absolutely vital to anyone who wants to do business in Europe. One of the key tasks that faced the EU member countries and the European Economic Area (EEA) was the requirement to bring several similar yet distinct standards together into one common standard for all members. The primary purpose of a single standard was to make it easier to sell and transport goods between the various countries and to maintain a safe working and living environment. The Directives that resulted from this merging of standards are now legal requirements for doing business in Europe. Products that meet these Directives are required to have a CE mark to signify compliance.

Member Countries

As of January 1, 2007, the members of the EU are Austria, Belgium, Bulgaria, Cyprus, Cxech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Ireland, Italy, Latvia, Lithonia, Luxembourg, Malta, Netherlands, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, and the United Kingdom. Iceland, Liechtenstein, and Norway together with the EU members make up the European Economic Area (EEA) and all are covered by the Directives.

Applicable Directives

There are several Directives that apply to our products. Directives may be amended, or added, as required.

- **Electromagnetic Compatibility Directive (EMC)** — this Directive attempts to ensure that products placed on the market do not generate electromagnetic disturbances that would affect other apparatus, including radio and/or telecommunications equipment.
- **Machinery Safety Directive** — this Directive covers the safety aspects of the equipment, installation, etc. There are several areas involved, including testing standards covering both electrical noise immunity and noise generation.
- **Low Voltage Directive** — this Directive is also safety related and covers electrical equipment that has voltage ranges of 50–1000VAC and/or 75–1500VDC.
- **Battery Directive** — this Directive covers the production, recycling, and disposal of batteries.

Compliance

Certain standards within each Directive already require mandatory compliance. The EMC Directive, which has gained the most attention, became mandatory as of January 1, 1996. The Low Voltage Directive became mandatory as of January 1, 1997.

Ultimately, we are all responsible for our various pieces of the puzzle. As manufacturers, we must test our products and document any test results and/or installation procedures that are necessary to comply with the Directives. As a machine builder, you are responsible for installing the products in a manner which will ensure compliance is maintained. You are also responsible for testing any combination of products that may (or may not) comply with the Directives when used together.

The end user of the products must comply with any Directives that may cover maintenance, disposal, etc. of equipment or various components. *Although we strive to provide the best assistance available, it is impossible for us to test all possible configurations of our products with respect to any specific Directive. Because of this, it is ultimately your responsibility to ensure that your machinery (as a whole) complies with these Directives and to keep up with applicable Directives and/or practices that are required for compliance.*

As of August 1, 2002, the DL105 (F1-130DR-CE, F1-130DD-CE, F1-130DR-D, and F1-130DD-D versions only), DL05, DL06, DL205, DL305, and DL405 PLC systems manufactured by Koyo Electronics Industries, FACTS Engineering or HOST Engineering when properly installed and used, conform to the Electromagnetic Compatibility (EMC), Low Voltage Directive, and Machinery Directive requirements of the following standards.

- **EMC Directive Standards Relevent to PLCs**
  EN50081-1 Generic emission standard for residential, commercial and light industry.
  EN50081-2 Generic emission standard for industrial environment.
  EN50082-1 Generic immunity standard for residential, commercial and light industry.
  EN50082-2 Generic immunity standard for industrial environment.
- **Low Voltage Directive Standard Applicable to PLCs**
  EN61010-1 Safety requirements for electrical equipment for measurement, control and laboratory use.
- **Product Specific Standard for PLCs**
  EN61131-2 Programmable controllers, equipment requirements and tests. This standard replaces the above generic standards for immunity and safety. However, the generic emissions standards must still be used in conjuction with the following standards:
  -EN61000-3-2 Harmonics
  -EN61000-3-2 Fluctuations
- **Warning on Electrostatic Discharge (ESD)**
  We recommend that all personnel take necessary precautions to avoid the risk of transferring static charges to inside the control cabinet, and clear warnings and instructions should be provided on the cabinet exterior. Such precautions may include, the use of earth straps, similar devices or the powering off of the equipment inside the enclosure before the door is opened.
- **Warning on Radio Interference (RFI)**
  This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

General Safety

- External switches, circuit breaker or external fusing, are required for these devices.
- The switch or circuit breaker should be mounted near the PLC equipment.

AutomationDirect is currently in the process of changing their testing procedures form the generic standards to the product specific standards.

Special Installation Manual

The installation requirements to comply with the requirements of the Machinery Directive, EMC Directive and Low Voltage Directive are slightly more complex than the normal installation requirements found in the United States. To help with this, we have published a special manual which you can order:

- **DA-EU-M** - EU Installation Manual that covers special installation requirements to meet the EU Directive requirements. Order this manual to obtain the most up-to-date information.

Other Sources of Information

Although the EMC Directive gets the most attention, other basic Directives, such as the Machinery Directive and the Low Voltage Directive, also place restrictions on the control panel builder. Because of these additional requirements it is recommended that the following publications be purchased and used as guidelines:

- BSI publication TH 42073: February 1996 - covers the safety and electrical aspects of the Machinery Directive
- EN 60204-1:1992 - General electrical requirements for machinery, including Low Voltage and EMC considerations
- IEC 1000-5-2: EMC earthing and cabling requirements
- IEC 1000-5-1: EMC general considerations

It may be possible for you to obtain this information locally; however, the official source of applicable Directives and related standards is:

The Office for Official Publications of the European Communities
L-2985 Luxembourg; quickest contact is via the World Wide Web at http://euro-op.eu.int/indexn.htm

Another source is:

British Standards Institution - Sales Department
Linford Wood
Milton Keynes
MK14 6LE
United Kingdom; the quickest contact is via the World Wide Web at http://www.bsi.org.uk

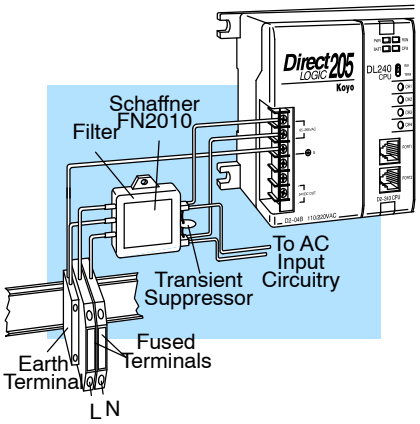
Basic EMC Installation Guidelines

Enclosures

The simplest way to meet the safety requirements of the Machinery and Low Voltage Directives is to house all control equipment in an industry standard lockable steel enclosure. This normally has an added benefit because it will also help ensure that the EMC characteristics are well within the requirements of the EMC Directive. Although the RF emissions from the PLC equipment, when measured in the open air, are well below the EMC Directive limits, certain configurations can increase emission levels. Holes in the enclosure, for the passage of cables or to mount operator interfaces, will often increase emissions.

AC Mains Filters

DL105, DL205 and DL305 AC powered base power supplies require extra mains filtering to comply with the EMC Directive on conducted RF emissions. All PLC equipment has been tested with filters from Schaffner, which reduce emissions to negligible levels if the filters are properly grounded (earth ground). A filter with a current rating suitable to supply all PLC power supplies and AC input modules should be selected. We suggest the FN2010 for DL105/DL205 systems and the FN2080 for DL305 systems. DL405 systems do not require extra filtering.





NOTE: Very few mains filters can reduce problem emissions to neglegible levels. In some cases, filters may increase conducted emissions if not properly matched to the problem emissions.

Suppression and Fusing

In order to comply with the fire risk requirements of the Low Voltage and Machinery Directive electrical standards EN 61010-1, and EN 60204-1, by limiting the power into "unlimited" mains circuits with power leads reversed, it is necessary to fuse both AC and DC supply inputs. You should also install a transient voltage suppressor across the power input connections of the PLC. Choose a suppressor such as a metal oxide varistor, with a rating of 275VAC working voltage for 230V nominal supplies (150VAC working voltage for 115V supplies) and high energy capacity (eg. 140 joules).

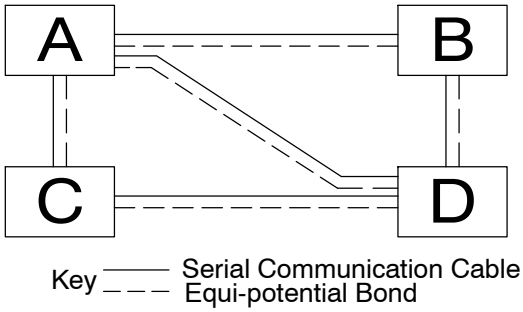
Transient suppressors must be protected by fuses and the capacity of the transient suppressor must be greater than the blow characteristics of the fuses or circuit breakers to avoid a fire risk. A recommended AC supply input arrangement for Koyo PLCs is to use twin 3 amp TT fused terminals with fuse blown indication, such as DINnectors DN-F10L terminals, or twin circuit breakers, wired to a Schaffner FN2010 filter or equivalent, with high energy transient suppressor soldered directly across the output terminals of the filter. PLC system inputs should also be protected from voltage impulses by deriving their power from the same fused, filtered, and surge-suppressed supply.

Internal Enclosure Grounding

A heavy-duty star earth terminal block should be provided in every cubicle for the connection of all earth ground straps, protective earth ground connections, mains filter earth ground wires, and mechanical assembly earth ground connections. This should be installed to comply with safety and EMC requirements, local standards, and the requirements found in IEC 1000–5–2.The Machinery Directive also requires that the common terminals of PLC input modules, and common supply side of loads driven from PLC output modules should be connected to the protective earth ground terminal.

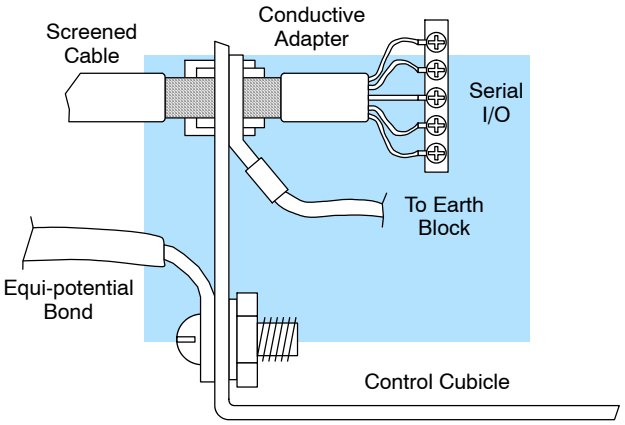
Equi-potential Grounding

Adequate site earth grounding must be provided for equipment containing modern electronic circuitry. The use of isolated earth electrodes for electronic systems is forbidden in some countries. Make sure you check any requirements for your particular destination. IEC 1000–5–2 covers equi-potential bonding of earth grids adequately, but special attention should be given to apparatus and control cubicles that contain I/O devices, remote I/O racks, or have inter-system communications with the primary PLC system enclosure. An equi-potential bond wire must be provided alongside all serial communications cables, and to any separate items of the plant which contain I/O devices connected to the PLC. The diagram shows an example of four physical locations connected by a communications cable.



Communications and Shielded Cables

Good quality 24 AWG minimum twisted-pair shielded cables, with overall foil and braid shields are recommended for analog cabling and communications cabling outside of the PLC enclosure. To date it has been a common practice to only provide an earth ground for one end of the cable shield in order to minimize the risk of noise caused by earth ground loop currents between apparatus. The procedure of only grounding one end, which primarily originated as a result of trying to reduce hum in audio systems, is no longer applicable to the complex industrial environment. Shielded cables are also efficient emitters of RF noise from the PLC system, and can interact in a parasitic manner in networks and between multiple sources of interference.



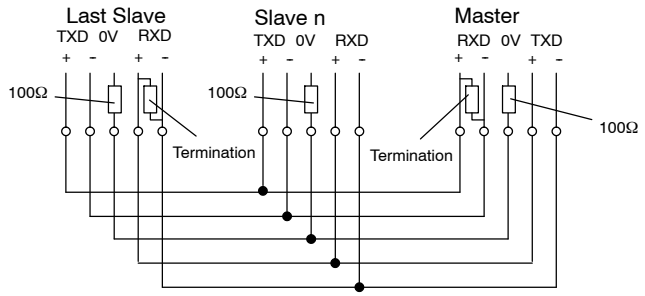
The recommendation is to use shielded cables as electrostatic "pipes" between apparatus and systems, and to run heavy gauge equi-potential bond wires alongside all shielded cables. When a shielded cable runs through the metallic wall of an enclosure or machine, it is recommended in IEC 1000-5-2 that the shield should be connected over its full perimeter to the wall, preferably using a conducting adapter, and not via a pigtail wire connection to an earth ground bolt. Shields must be connected to every enclosure wall or machine cover that they pass through.

Analog and RS232 Cables

Providing an earth ground for both ends of the shield for analog circuits provides the perfect electrical environment for the twisted pair cable as the loop consists of signal and return, in a perfectly balanced circuit arrangement, with connection to the common of the input circuitry made at the module terminals. RS232 cables are handled in the same way.

Multidrop Cables

RS422 twin twisted pair, and RS485 single twisted pair cables also require a 0V link, which has often been provided in the past by the cable shield. It is now recommended that you use triple twisted pair cabling for RS422 links, and twin twisted pair cable for RS485 links. This is because the extra pair can be used as the 0V inter-system link. With loop DC power supplies earth grounded in both systems, earth loops are created in this manner via the inter-system 0v link. The installation guides encourage earth loops, which are maintained at a low impedance by using heavy equi-potential bond wires. **To account for non-European installations using single-end earth grounds, and sites with far from ideal earth ground characteristics, we recommend the addition of 100 ohm resistors at each 0V link connection in network and communications cables.**



Shielded Cables within Enclosures

When you run cables between PLC items within an enclosure which also contains susceptible electronic equipment from other manufacturers, remember that these cables may be a source of RF emissions. There are ways to minimize this risk. Standard data cables connecting PLCs and/or operator interfaces should be routed well away from other equipment and their associated cabling. You can make special serial cables where the cable shield is connected to the enclosure's earth ground at both ends, the same way as external cables are connected.

Network Isolation

For safety reasons, it is a specific requirement of the Machinery Directive that a keyswitch must be provided that isolates any network input signal during maintenance, so that remote commands cannot be received that could result in the operation of the machinery. The FA-ISONET does not have a keyswitch! Use a keylock and switch on your enclosure which when open removes power from the FA-ISONET. To avoid the introduction of noise into the system, any keyswitch assembly should be housed in its own earth grounded steel box and the integrity of the shielded cable must be maintained.

Again, for further information on EU directives we recommend that you get a copy of our EU Installation Manual (DA-EU-M). Also, if you are connected to the World Wide Web, you can check the EU Commision's official site at: http://eur-op.eu.int/

Numbering Systems

G

In This Appendix. . . .

— Introduction
— Binary Numbering System
— Hexadecimal Numbering System
— Octal Numbering System
— Binary Coded Decimal (BCD) Numbering System
— Real (Floating Point) Numbering System
— BCD/Binary/Decimal/Hex/Octal - What is the Difference?
— Data Type Mismatch
— Signed vs. Unsigned Integers
— AutomationDirect.com Products and Data Types

Introduction

As almost anyone who uses a computer is somewhat aware, the actual operations of a computer are done with a binary number system. Traditionally, the two possible states for a binary system are represented by the digits for ”zero” (0) and ”one” (1) although ”off” and ”on” or sometimes ”no” and yes” are closer to what is actually involved. Most of the time a typical PC user has no need to think about this aspect of computers, but every now and then one gets confronted with the underlying nature of the binary system.

A PLC user should be more aware of the binary system specifically the PLC programmer. This appendix will provide an explanation of the numbering systems most commonly used by a PLC.

Binary Numbering System

Computers, including PLCs, use the Base 2 numbering system, which is called Binary or Boolean. Like in a computer there are only two valid digits in Base 2 a PLC relies on, zero and one, or off and on respectively. You would think that it would be hard to have a numbering system built on Base 2 with only two possible values, but the secret is by encoding using several digits.

Each digit in the base 2 system when referenced by a computer is called a bit. When four bits are grouped together, they form what is known as a nibble. Eight bits or two nibbles would be a byte. Sixteen bits or two bytes would be a word as in. Thirty-two bits or two words is a double word as in Table 1.

Double Word																															
Word																**Word**															
Byte								**Byte**								**Byte**								**Byte**							
Nibble				Nibble				Nibble				Nibble				Nibble				Nibble				Nibble				Nibble			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1

Binary is not “natural“ for us to use since we have grown up using the base 10 system. Base 10 uses the numbers 0-9, as we are all well aware. From now on, the different bases will be shown as a subscripted number following the number. Example: 10 decimal would be 10_{10}.

Table 2 shows how base 2 numbers relate to their decimal equivalents.

A nibble of 1001_2 would be equal to a decimal number 9 ($1*2^3 + 1*2^0$ or $8_{10} + 1_{10}$). A byte of 11010101_2 would be equal to 213 ($1*2^7 + 1*2^6 + 1*2^4 + 1*2^2 + 1*2^0$ or $128_{10} + 64_{10} + 16_{10} + 4_{10} + 1_{10}$).

Binary/Decimal Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Power	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal Bit Value	32678	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Max Value	65535_{10}															

Table 2

Hexadecimal Numbering System

The binary numbering system can be difficult and cumbersome to interpret for some users. Therefore, the hexadecimal numbering system was developed as a convenience for humans since the PLC (computer) only understands pure binary. The hexadecimal system is useful because it can represent every byte (8 bits) as two consecutive hexadecimal digits. It is easier for us to read hexadecimal numbers than binary numbers.

The hexadecimal numbering system uses 16 characters (base 16) to represent values. The first ten characters are the same as our decimal system, 0–9, and the first six letters of the alphabet, A–F. Table 3 lists the first eighteen decimal numbers; 0–17 in the left column and the equivalent hexadecimal numbers are shown in the right column.

Decimal	Hex	Decimal	Hex
0	0	9	9
1	1	10	A
2	2	11	B
3	3	12	C
4	4	13	D
5	5	14	E
6	6	15	F
7	7	16	10
8	8	17	11

Table 3

Note that “10” and “11“ in hex are not the same as “10“ and “11“ in decimal. Only the first ten numbers 0–9 are the same in the two representations. For example, consider the hex number “D8AF“. To evaluate this hex number use the same method used to write decimal numbers. Each digit in a decimal number represents a multiple of a power of ten (base 10). Powers of ten increase from right to left. For example, the decimal number 365 means $3x10^2 + 6x10 + 5$. In hex each digit represents a multiple of a power of sixteen (base 16). Therefore, the hex number D8AF translated to decimal means $13x16^3 + 8x16^2 + 10x16 + 15 = 55471$. However, going through the arithmetic for hex numbers in order to evaluate them is not really necessary. The easier way is to use the calculator that comes as an accessory in Windows. It can convert between decimal and hex when in “Scientific“ view.

Note that a hex number such as “365“ is not the same as the decimal number “365“. Its actual value in decimal terms is $3x16^2$ 6x16 + 5 = 869. To avoid confusion, hex numbers are often labeled or tagged so that their meaning is clear. One method of tagging hex numbers is to append a lower case “h“ at the end. Another method of labeling is to precede the number with 0x. Thus, the hex number “D8AF“ can also be written “D8AFh“, where the lower case “h” at the end is just a label to make sure we know that it is a hex number. Also, D8AF can be written with a labeling prefix as “0xD8AF”.

Octal Numbering System

Many of the early computers used the octal numbering system for compiled printouts. Today, the PLC is about the only device that uses the Octal numbering system. The octal numbering system uses 8 values to represent numbers. The values are 0–7 being Base 8. Table 4 shows the first 32 decimal digits in octal. Note that the octal values are 0–7, 10–17, 20–27, and 30–37.

Octal	Decimal	Octal	Decimal
0	0	20	16
1	1	21	17
2	2	22	18
3	3	23	19
4	4	24	20
5	5	25	21
6	6	26	22
7	7	27	23
10	8	30	24
11	9	31	25
12	10	32	26
13	11	33	27
14	12	34	28
15	13	35	29
16	14	36	30
17	15	37	31

Table 4

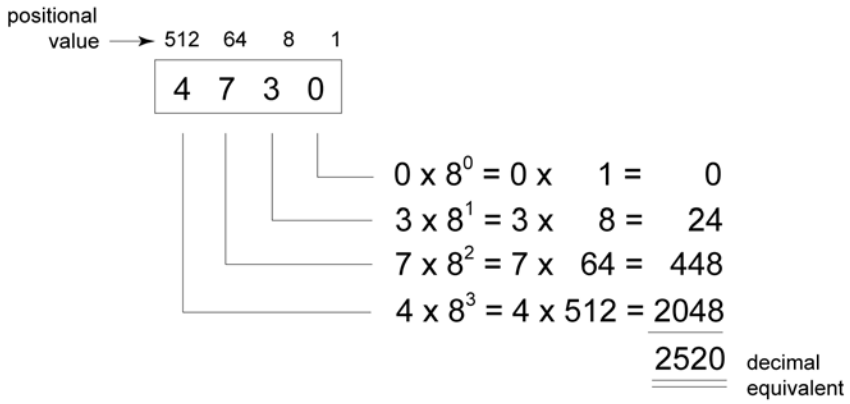
This follows the ***Direct***LOGIC PLCs. Refer to the bit maps in Chapter 4 and notice that the memory addresses are numbered in octal, as well as each bit. The octal system is much like counting in the decimal system without the digits 8 and 9 being available.

The general format for four digits of the octal number is:

$$(d \times 8^0) + (d \times 8^1) + (d \times 8^2) + (d \times 8^3)$$

where “d“ means digit. This is the same format used in the binary, decimal, or hexadecimal systems except that the base number for octal is 8.

Using the powers of expansion, the example below shows octal 4730 converted to decimal.



Binary Coded Decimal (BCD) Numbering System

BCD is a numbering system where four bits are used to represent each decimal digit. The binary codes corresponding to the hexadecimal digits A–F are not used in the BCD system. For this reason numbers cannot be coded as efficiently using the BCD system. For example, a byte can represent a maximum of 256 different numbers (i.e. 0–255) using normal binary, whereas only 100 distinct numbers (i.e. 0–99) could be coded using BCD. Also, note that BCD is a subset of hexadecimal and neither one does negative numbers.

BCD Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Power	10^3				10^2				10^1				10^0			
Bit Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Max Value	9				9				9				9			

Table 5

One plus for BCD is that it reads like a decimal number, whereas 867 in BCD would mean 867 decimal. No conversion is needed.

Real (Floating Point) Numbering System

The terms Real and floating-point both describe IEEE-754 floating point arithmetic. This standard specifies how single precision (32 bit) and double precision (64 bit) floating point numbers are to be represented as well as how arithmetic should be carried out on them. Most PLCs use the 32-bit format for floating point (or Real) numbers which will be discussed here.

Real (Floating Point 32) Bit Pattern																
Bit #	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Sign	Exponent								Mantissa						
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Mantissa (continues from above)															

Table 6

Floating point numbers, which ***Direct***LOGIC PLCs use, have three basic components: sign, exponent and mantissa. The 32 bit word required for the IEEE standard floating point numbers is shown in Table 6. It is represented as a number bits from 0 to 31, left to right. The first bit (31) is the sign bit, the next eight bits (30-23) are the exponent bits and the final 23 bits (22-0) are the fraction bits. In summary:

The sign bit is either “0” for positive or “1“ for negative;

The exponent uses base 2;

The first bit of the mantissa is typically assumed to be “1.fff“, where “f“ is the field of fraction bits.

The Internet can provide a more in-depth explanation of the floating point numbering system. One website to look at is:

http://www.psc.edu/general/software/packages/ieee/ieee.php

BCD/Binary/Decimal/Hex/Octal - What is the Difference?

Sometimes there is confusion about the differences between the data types used in ***Direct***LOGIC PLCs. The PLC's native data format is BCD, while the I/O numbering system is octal. Other numbering formats used are binary, decimal and Real. Although data is stored in the same manner (0's and 1's), there are differences in the way that the PLC interprets it.

While all of the formats rely in the base 2 numbering system and bit-coded data, the format of the data is dissimilar. The formats discussed are shown in Table 7 below.

Binary/Decimal Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Bit Value	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Max Value	65535															
Hexidecimal Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Bit Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Max Value	F				F				F				F			
BCD Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Bit Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Max Value	9				9				9				9			
Octal Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Value	1	4	2	1	4	2	1	4	2	1	4	2	1	4	2	1
Max Value	1	7			7			7			7			7		
Octal Bit Pattern																
Bit #	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Sign	Exponent								Mantissa						
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Mantissa (continued from above)															

Table 7

As seen in Table 7, the BCD and hexadecimal formats are similar, although the maximum number for each grouping is different (9 for BCD and F for hexadecimal). This allows both formats to use the same display method. The unfortunate side effect is that unless the data type is documented, it's difficult to know what the data type is unless it contains the letters A-F.

Data Type Mismatch

Data type mismatching is a common problem when using an operator interface. Diagnosing it can be a challenge until you identify the symptoms. Since the PLC uses BCD as the native format, many people tend to think it is interchangeable with binary (unsigned integer) format. This is true to some extent, but not in this case. Table 8 shows how BCD and binary numbers differ.

Data Type Mismatch												
Decimal	0	1	2	3	4	5	6	7	8	9	10	11
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	0001 0000	0001 0001
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	0000 1010	0000 1011

Table 8

As the table shows, BCD and binary share the same bit pattern up until you get to the decimal number 10. Once you get past 10, the bit pattern changes. The BCD bit pattern for the decimal 10 is actually equal to a value of 16 in binary, causing the number to jump six digits by when viewing it as the BCD. With larger numbers, the error multiplies. Binary values from 10 to 15 Decimal are actually invalid for the BCD data type.

Looking at a larger number, such as the value shown in Table 9, both the BCD bit pattern and the decimal bit pattern correspond to a base 10 value of 409510. If bit patterns are read, or interpreted, in a different format than what is used to write them, the data will not be correct. For instance, if the BCD bit pattern is interpreted as a decimal (binary) bit pattern, the result is a base 10 value of 1653310. Similarly, if you try to view the decimal (binary) bit pattern as a BCD value, it is not a valid BCD value at all, but could be represented in hexadecimal as 0xFFF.

Base 10 Value	BCD Bit Pattern	Binary Bit Pattern
4095	0100 0000 1001 0101	1111 1111 1111

Table 9

Look at the following examples and note the same value represented by the different numbering systems. The decimal values of 67 and 4660 are used.

Decimal	6	7			Decimal	4	6	6	0
BCD	0110	0111			BCD	0100	0110	0110	0000
Binary	0100	0011			Binary	0001	0010	0011	0100
Hex	4	3			Hex	1	2	3	4
Octal	1	0	3		Octal	1	1	0	6	4

Signed vs. Unsigned Integers

So far, we have dealt with unsigned data types only. Now we will deal with signed data types (negative numbers). The BCD and hexadecimal numbering systems do not use signed data types.

In order to signify that a number is negative or positive, we must assign a bit to it. Usually, this is the Most Significant Bit (MSB) as shown in Table 10. For a 16-bit number, this is bit 15. This means that for 16-bit numbers we have a range of -32767 to 32767.

Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 10

There are two ways to encode a negative number: two's complement and Magnitude Plus sign. The two methods are not compatible.

The simplest method to represent a negative number is to use one bit of the PLC word as the sign of a number while the remainder of the word gives its magnitude. It is general convention to use the most significant bit (MSB) as the sign bit: a 1 will indicate a negative, and a 0 a positive number. Thus, a 16 bit word allows numbers in the range ±32767. The following tables show a representation of 100 and a representation of -100 in this format.

Magnitude Plus Sign	
Decimal	**Binary**
100	0000 0000 0110 0100
-100	1000 0000 0110 0100

Table 11

Two's complement is a bit more complicated. Without getting involved with a full explanation, a simple formula for two's complement is to invert the binary and add one (see Table 12). Basically, 1's are being changed to 0's and all 0's are being changed to 1, then a 1 is added.

Two's Compliment	
Decimal	**Binary**
100	0000 0000 0110 0100
-100	1111 1111 1001 1100

Table 12

More information about 2's complement can be found on the Internet, however most of the websites deal with 8-bit examples. Two fairly good websites are listed below, you can also do a search and checkout more websites.

http://www.evergreen.edu/biophysics/technotes/program/2s_comp.htm

http://www.website.masmforum.com/tutorials/fptute/fpuchap2.htm

AutomationDirect.com Products and Data Types

_Direct_LOGIC PLCs

The ***Direct***LOGIC PLC family uses the octal numbering system for all addressing which includes: inputs, outputs, internal V-memory locations, timers, counters, internal control relays (bits), etc. Most data in the PLC, including timer and counter current values, is in BCD format by default. User data in V-memory locations may be stored in other data types if it is changed by the programmer, or comes from some external source, such as an operator interface. Any manipulation of data must use instructions appropriate for that data type which includes: Load instructions, Math instructions, Out box instructions, comparison instructions, etc. In many cases, the data can be changed from one data type to another, but be aware of the limitations of the various data types when doing so. For example, to change a value from BCD to decimal (binary), use a BIN instruction box. To change from BCD to a real number, use a BIN and a BTOR instruction box. When using Math instructions, ***the data types must match***. For example, a BCD or decimal (binary) number cannot be added to a real number, and a BCD number cannot be added to a decimal (binary) number. If the data types are mismatched, the results of any math operation will be meaningless.

When using the Data View in ***Direct***SOFT, be certain that the proper format is selected for the element to be viewed. The data type and length is selected using the drop-down boxes at the top of the Data View window. Also notice that BCD is called BCD/Hex. Remember that BCD is a subset of hexadecimal so they share a display format even though the values may be different. This is where good documentation of the data type stored in memory is crucial.

C-more/C-more Micro-Graphic Panels

In the C-more and C-more Micro-Graphic HMI operator panels, the 16-bit BCD format is listed as “BCD int 16“. Binary format is either “Unsigned int 16“ or “Signed int 16“ depending on whether or not the value can be negative. Real number format is “Floating PT 32”.

More available formats are, “BCD int 32“, “Unsigned int 32“ and “Signed int 32“.

Index

A

Accumulating Fast Timer instruction, 5–30

Accumulating Timer instruction, 5–30

Add Double instruction, 5–61

Add instruction, 5–60

Agency approvals, 2–9

And Double instruction, 5–53

And If Equal instruction, 5–19

And If Not Equal instruction, 5–19

And Immediate instruction, 5–24

And instruction, 5–11, 5–22, 5–52

And Not Immediate instruction, 5–24

And Not instruction, 5–11, 5–17

And Store instruction, 5–12

ASCII Constant instruction, 5–84

Auxiliary functions, 4–8, A–2

B

BCD numbers, 4–21

Binary Coded Decimal instruction, 5–72

Binary instruction, 5–71

C

Cables
- operator interfaces, 2–17
- programming, 1–11
- programming devices, 2–17, 4–5

Common terminals, 2–19

Communications port, 4–4

Compare instruction, 5–58

Connectors
- common terminals, 2–19
- removal, 2–6

Counter instruction, 5–33

CPU
- configuration, A–5
- indicators, 8–6
- instruction list, 5–2
- memory map, 4–22
- modes, 4–6

CPU specifications, 4–3

D

Data Label instruction, 5–84

Diagnostics, 4–14, 8–2

DIN rail mounting, 2–10

Disable Interrupts instruction, 5–80

Divide instruction, 5–65

DL105 Micro PLC
- front panel, 2–5
- mounting guidelines, 2–7

DL105 Aliases, 4–28

Drum instruction, 6–2
- drum control techniques, 6–10
- EDRUM (Event Drum), 6–12
- handheld programmer mnemonics, 6–14
- overview of drum operation, 6–8
- step transition, 6–4

Drum sequencer programming, 1–11

E

Emergency stop, 2–3

Enable Interrupts instruction, 5–80
Encode instruction, 5–69
Encoder signals, 3–17
End instruction, 5–3
Environmental specifications, 2–9
Equal relays, 3–9, D–3
Error codes
  code locations, 8–3
  listing, B–2
  pulse output errors, 3–41
Exclusive Or Double instruction, 5–57
Exclusive Or instruction, 5–56

F

Fault instruction, 5–83
Forced I/O, 4–13

H

Handheld programmer, A–6
  EEPROM operations, A–7
Hexadecimal numbers, 4–21
High–speed I/O
  discrete inputs with filter, 3–51
  features, 3–2
  high–speed counter, 3–6
  high–speed interrupts, 3–43
  modes, 3–4
  pulse catch input, 3–48
  pulse output, 3–23
  quadrature counter, 3–17
Home search profile, 3–36

I

I/O response time, 4–15
I/O Type Selection, 1–5
Initial Stage instruction, 7–20
Initial Stages, 7–5
Input simulator, 1–7, 2–28
Installation
  grounding, 2–8
  panel design specifications, 2–8
Instructions
  accumulator / stack Load, 5–40
  bit operations, 5–67
  boolean, 5–3, 5–9
  comparative boolean, 5–17
  drum, 6–2
  execution times, 4–19, C–2
  immediate, 5–23
  interrupt, 5–80
  list of, 5–2
  logical, 5–52
  math, 5–60
  message, 5–83
  number conversions, 5–71
  program control, 5–78
  program control instructions, 4–19
  stage, 7–19
  stage programming, 7–2
  table, 5–74
  timer, counter, and shift register, 5–27
Interrupt instruction, 5–80
Interrupt Return instruction, 5–80
Interrupts
  external, 3–45
  HSIO input, 3–43
  timed, 3–45

J

Jump instruction, 7–7, 7–20

L

Load Address instruction, 5–48
Load Double instruction, 5–46
Load instruction, 5–45
Load Label instruction, 5–75

M

Maintenance, 8–2
Manual, organization, 1–2
Master Line Reset instruction, 5–78
Master Line Set instruction, 5–78
Memory map, 4–22, 4–26
Message instructions, 5–83
Motion control profile, 3–23

Mounting, Guidelines, 2–7
Move instruction, 5–74
Move Memory Cartridge instruction, 5–75
Multiply instruction, 5–64

N

Numbering Systems
- BCD, G–5
- Binary, G–2
- Floating Point, G–6
- Hexadecimal, G–3
- Octal, G–4

Numerical Constant instruction, 5–84

O

Octal numbers, 4–20, 4–22
Or Double instruction, 5–55
Or If Equal instruction, 5–18
Or If Not Equal instruction, 5–18
Or Immediate instruction, 5–23
Or instruction, 5–10, 5–21, 5–54
Or Not Immediate instruction, 5–23
Or Not instruction, 5–10, 5–21
Or Out Immediate instruction, 5–25
Or Store instruction, 5–12
Out Double instruction, 5–49
Out Formatted instruction, 5–50
Out instruction, 5–49

P

Panel layout, 2–8
Part Numbers, 1–5
Password, 4–10, A–8
Pause instruction, 5–16
Pop instruction, 5–50
Positive Differential instruction, 5–15
Power wiring, 1–8
Presets, 3–8, 3–10
Program control instructions, 5–78
Program mode, 4–12
Programming, concepts, 1–11
Programming methods, 1–8

Q

Quadrature counter, 3–17
Quick start, 1–6

R

Registration profile, 3–29, 3–33
Relay wiring, 2–22, 2–24
Reset instruction, 5–15
Retentive memory, 4–9
Run mode, 4–12
Run time edits, 8–14

S

Safety, panel design specifications, 2–4
Safety guidelines, 2–2
Scan time, 4–18
Scratchpad memory, 1–9, 4–9
Set instruction, 5–15
Shift Left instruction, 5–67
Shift Right instruction, 5–68
Sinking / sourcing I/O, 2–20
Special relays, 4–13, 4–25, D–2, E–2
Specifications
- CPU, 4–3
- environmental, 2–9
- F1–130AA, 2–38
- F1–130AD, 2–34
- F1–130DA, 2–40
- F1–130DD, 2–36
- F1–130DD–D, 2–44
- F1–130DR, 2–32
- F1–130DR–D, 2–42
- motion profiles, 3–26
- panel design, 2–8

Stack, 5–7
Stage Counter instruction, 5–35
Stage instructions, 7–19

Stage programming, 1–11, 7–2
four steps to writing a stage program, 7–9
garage door opener example, 7–10
initial stages, 7–5
introduction, 7–2
jump instruction, 7–7
mutually exclusive transitions, 7–14
parallel processes, 7–12
power flow transition, 7–18
program organization, 7–15
questions and answers, 7–21
stage data type, 4–25
stage instruction characteristics, 7–6
stage view, 7–18
state transition diagrams, 7–3
supervisor process, 7–17
timer inside stage, 7–13
Standard RLL Programming, 1–11
Store If Equal instruction, 5–17
Store If Not Equal instruction, 5–17
Store immediate instruction, 5–23
Store instruction, 5–9, 5–20
Store Not Immediate instruction, 5–23
Store Not instruction, 5–9, 5–20
Subtract double instruction, 5–63
Subtract instruction, 5–62
System, panel design specifications, 2–4
System design steps, 1–10

T

Table instructions, 5–74
Technical support, 1–2, 1–4
Trapezoidal profile, 3–30
Troubleshooting, 8–2
Troubleshooting, 8–2
counter input, 3–7
electrical noise, 8–10
error codes, B–2
!/O points, 8–8
program debug, 8–11
Troubleshooting guide
HSIO Mode 10, 3–16
HSIO Mode 20, 3–22
HSIO Mode 30, 3–41

U

Up Down Counter instruction, 5–37

V

V–memory, 4–22, 4–26
Velocity profile, 3–29, 3–38

W

Wiring
counter input, 3–7
DC inputs, 2–25
DC outputs, 2–26
encoder, 3–18
high–speed I/O, 2–27
input simulator, 1–7, 2–28
power input, 1–7, 2–11, 2–14
pulse output, 3–25
relay outputs, 2–22
Wiring Guidelines, 2–11, 2–14

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP